A MARSHALL CAVENDISH 47 COMPUTER COURSE IN WEEKLY PARTS

INPUT

LEARN PROGRAMMING - FOR FUN AND THE FUTURE

UK £1.00 Republic of Ireland £1.25 Malta 85c Australia $2.25 New Zealand $2.95

INPUT

Vol. 4 **No 47**

INDEX

The last part of INPUT, Part 52, will contain a complete, cross-referenced index. For easy access to your growing collection, a cumulative index to the contents of each issue is contained on the inside back cover.

PICTURE CREDITS
Front cover, Graeme Harris. Pages 1461, 1464, Spectrum Colour Library/Berry Fallon Design. Pages 1462, 1463, Peter Reilly. Pages 1466, 1470, George Logan. Page 1475, Peter Reilly. Pages 1476, 1478, 1480, Gary Wing. Pages 1482, 1484, Graen. Harris. Pages 1487, 1488, 1491, 1492, Stuart Robertson.

Published by Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA, England. Typeset by MS Filmsetting Limited, Frome, Somerset. Printed by Cooper Clegg Web Offset Ltd, Gloucester and Howard Hunt Litho, London.

There are four binders each holding 13 issues.

HOW TO ORDER YOUR BINDERS

UK and Republic of Ireland:
Send £4.95 (inc p & p) (IR£5.95) for each binder to the address below:
Marshall Cavendish Services Ltd, Department 980, Newtown Road, Hove, Sussex BN3 7DN
Australia: See inserts for details, or write to INPUT, Times Consultants, PO Box 213, Alexandria, NSW 2015
New Zealand: See inserts for details, or write to INPUT, Gordon and Gotch (NZ) Ltd, PO Box 1595, Wellington
Malta: Binders are available from local newsagents.

BACK NUMBERS

Back numbers are supplied at the regular cover price (subject to availability).

UK and Republic of Ireland:
INPUT, Dept AN, Marshall Cavendish Services, Newtown Road, Hove BN3 7DN

Australia, New Zealand and Malta:
Back numbers are available through your local newsagent.

COPIES BY POST

Our Subscription Department can supply copies to any UK address regularly at £1.00 each. For example the cost of 26 issues is £26.00; for any other quantity simply multiply the number of issues required by £1.00. Send your order, with payment to:

Subscription Department, Marshall Cavendish Services Ltd, Newtown Road, Hove, Sussex BN3 7DN

Please state the title of the publication and the part from which you wish to start.

HOW TO PAY: Readers in UK and Republic of Ireland: All cheques or postal orders for binders, back numbers and copies by post should be made payable to:
Marshall Cavendish Partworks Ltd.

QUERIES: When writing in, please give the make and model of your computer, as well as the Part No., page and line where the program is rejected or where it does not work. We can only answer specific queries – and please do not telephone. Send your queries to INPUT Queries, Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA.

INPUT IS SPECIALLY DESIGNED FOR:

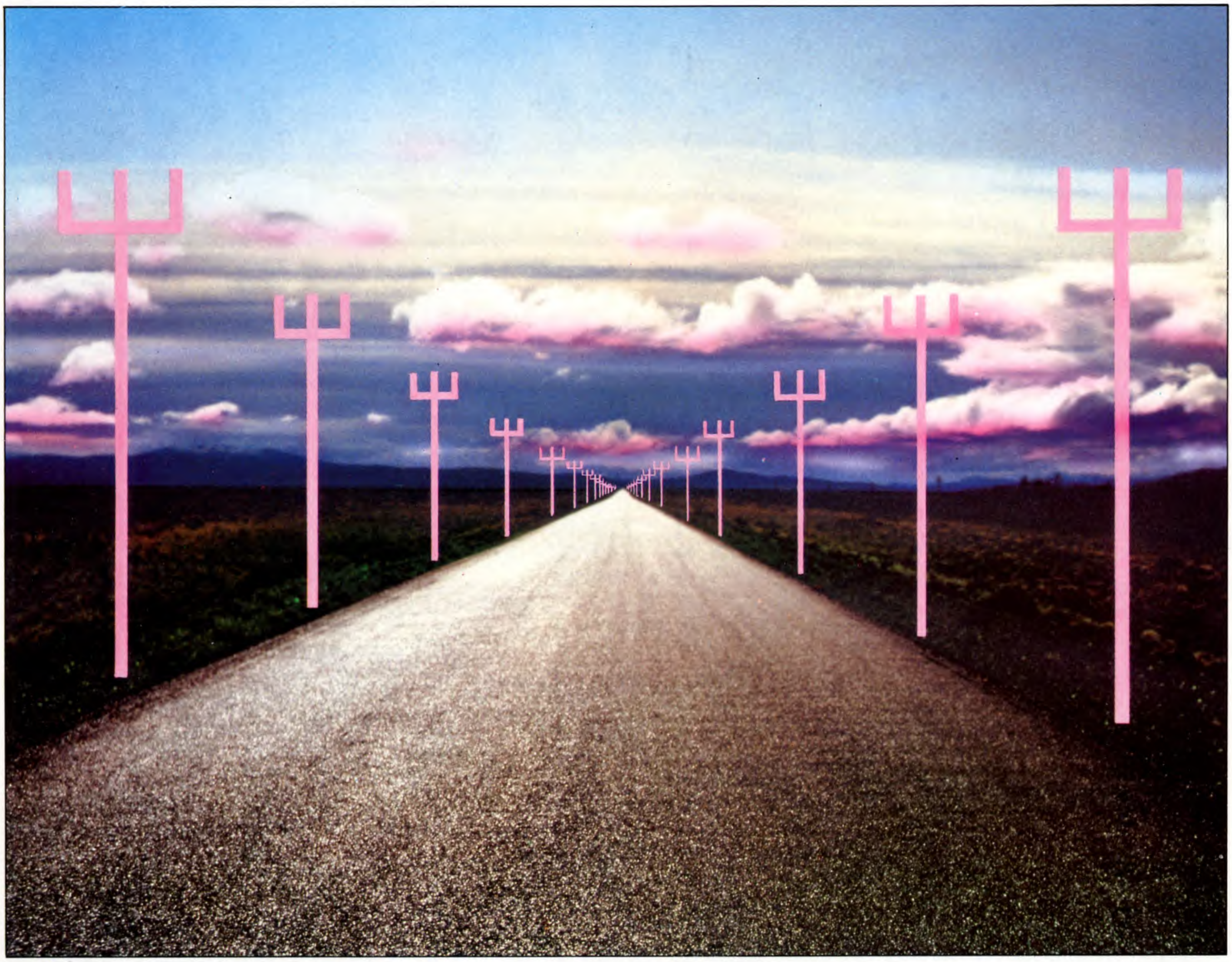
The SINCLAIR ZX SPECTRUM (16K, 48K, 128 and +), COMMODORE 64 and 128, ACORN ELECTRON, BBC B and B+, and the DRAGON 32 and 64.

In addition, many of the programs and explanations are also suitable for the SINCLAIR ZX81, COMMODORE VIC 20, and TANDY COLOUR COMPUTER in 32K with extended BASIC. Programs and text which are specifically for particular machines are indicated by the following symbols:

SPECTRUM 16K, 48K, 128, and +
COMMODORE 64 and 128
ACORN ELECTRON, BBC B and B+
DRAGON 32 and 64
ZX81
VIC 20
TANDY TRS80 COLOUR COMPUTER

ADDING SOME DEPTH

- PERSPECTIVE
- VANISHING POINT
- VIEWPOINT
- DIMINISHING SIZE
- SHADING



Add a feeling of depth and three-dimensionality to your graphics by incorporating the principles of perspective and adding a touch of shading

Most people think of perspective in drawing as common sense and tend to draw sketches with perspective built in. Surprisingly, perspective is not at all common sense, and the early artists had no idea of how to add depth to their pictures. In fact, it was only during the Renaissance, when painters attempted to represent the world in a realistic way that the rules of perspective were explored and formulated. These rules are the same as artists use today, and apply equally to pictures drawn by a computer.

The development of perspective took several centuries but the rules are not at all difficult to understand. Basically, all horizontal lines going from side to side of the picture stay horizontal, all vertical lines stay vertical, and all lines going 'into' the picture from front to back converge to a point in the middle of the picture, called the *vanishing point*. You've already met this idea in the article on wireframe drawings, which used the principle of perspective to draw the cube—see page 605.

Only parallel lines that are supposed to be perpendicular to the picture converge at the central *principal* vanishing point. Other sets of parallel lines converge on their own vanishing point to one side of the principal one. All vanishing points lie on the same line, though, called the horizon. It is called this because if you could look through the picture to open space this would correspond to the real horizon. However, do bear in mind that this is an imaginary line and needn't correspond to any real line in your picture.

These rules are obviously needed for realistic pictures, but on a simpler level they mean you can quickly create a feeling of depth simply by incorporating something like a road or river or track that vanishes into the distance. You can create an even stronger effect by using a grid of lines, a trick often used in futuristic or space scenes or in TV advertisements that use computer graphics.

PERSPECTIVE GRID

The first program draws two grids, one forming the 'ground' and the other the 'sky', with you, the viewer, seemingly suspended somewhere in between looking into infinity.

```
10 BORDER Ø: PAPER Ø: INK 7
20 CLS : INPUT "VANISHING FACTOR□";V
40 FOR K= -126 TO 127 STEP 6
50 LET Y=174: LET X=K+V*K: IF ABS
   (X)>127+(X<1) THEN LET X=SGN
   (X)*127-(X<1): LET
   Y=100+(X-K)*74/(V*K)
60 PLOT 127-K,100: DRAW 127-X-PEEK
   23677,Y-PEEK 23678
70 PLOT 127-K,74: DRAW 127-X-PEEK
   23677,174-Y-PEEK 23678
80 NEXT K
90 LET F=V↑(1/6): LET Y=F
100 LET Y=Y*F: IF Y>77 THEN GOTO 140
110 PLOT Ø,97+Y: DRAW 255,Ø
120 PLOT Ø,77-Y: DRAW 255,Ø
130 GOTO 100
140 IF INKEY$="" THEN GOTO 140
150 GOTO 20
```

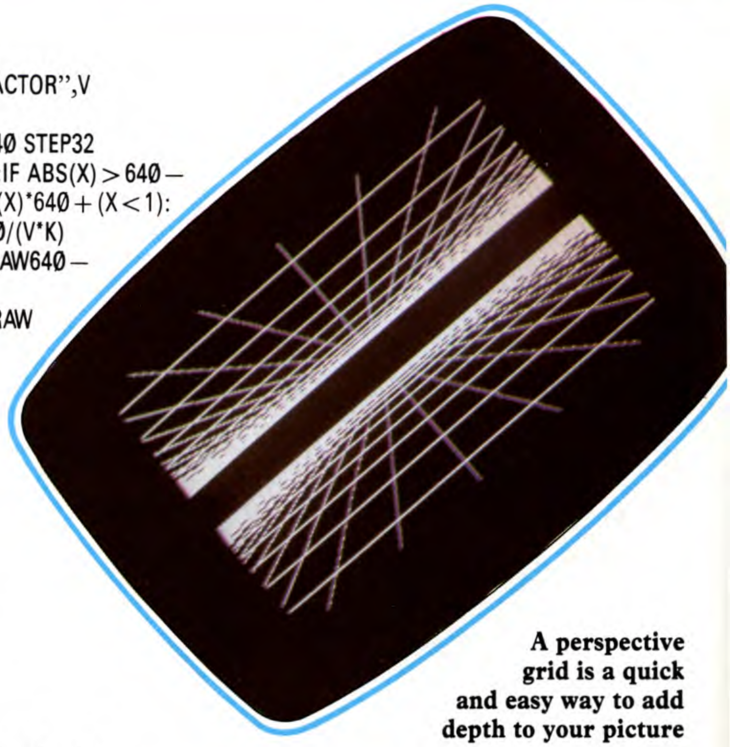
```
20 INPUT"□VANISHING FACTOR■";V
30 HIRES Ø,1
40 FOR K= -159 TO 159 STEP 6
50 Y=199:X=K+V*K:IF ABS(X)>16Ø
   -(X<1)THENX=SGN(X)*16Ø+(X<1):
   Y=111+(X-K)*8Ø/(V*K)
60 LINE 16Ø-K,111,16Ø-X,Y,1
70 LINE 16Ø-K,9Ø,16Ø-X,199-Y,1
80 NEXT K
90 F=V↑(1/6):Y=F
100 Y=Y*F:IF Y>93 THEN 14Ø
110 LINE Ø,1Ø8+Y,319,1Ø8+Y,1
120 LINE Ø,94-Y,319,94-Y,1
130 GOTO 1ØØ
140 GET A$:IF A$="" THEN 14Ø
150 NRM:GOTO 2Ø
```

```
10 MODEØ
20 INPUT"VANISHING FACTOR",V
30 CLS
40 FOR K= -64Ø TO 64Ø STEP32
50 Y=1Ø23:X=K+V*K:IF ABS(X)>64Ø-
   (X<1)THEN X=SGN(X)*64Ø+(X<1):
   Y=624+(X-K)*4ØØ/(V*K)
60 MOVE64Ø-K,624:DRAW64Ø-
   X,Y
70 MOVE64Ø-K,4ØØ:DRAW
   64Ø-X,1Ø23-Y
80 NEXT
90 F=V∧(1/6):Y=F
100 Y=Y*F:IFY>4ØØ
   THEN 14Ø
110 MOVEØ,62Ø+Y:
   DRAW128Ø,62Ø+Y
120 MOVEØ,4Ø4-Y:
   DRAW128Ø,4Ø4-Y
130 GOTO 1ØØ
140 D=GET:GOTO 2Ø
```

```
10 PMODE4,1
20 CLS:INPUT" VANISHING FACTOR
   □";V
30 PCLS:SCREEN1,1
40 FORK= -126TO127 STEP 6
50 Y=191:X=K+V*K:IFABS(X)>127-
   (X<1)THENX=SGN(X)*127+(X<1):
   Y=111+(X-K)*8Ø/(V*K)
60 LINE(127-K,111)-(127-X,Y),PSET
70 LINE(127-K,8Ø)-(127-X,191-Y),
   PSET
80 NEXT
90 F=V↑(1/6):Y=F
100 Y=Y*F:IFY>83 THEN14Ø
110 LINE(Ø,1Ø8+Y)-(255,1Ø8+Y),PSET
120 LINE(Ø,84-Y)-(255,84-Y),PSET
130 GOTO1ØØ
140 IF INKEY$="" THEN14Ø ELSE2Ø
```

By altering the position of the vanishing points you can create quite different effects. Enter a low value for V to start with, say 2. This makes the lines on the bottom grid converge to a point above the middle line (the horizon) and those on the top grid converge to a point below the horizon. It's this that gives you the impression of hovering between the two grids. You can dramatically alter the appearance of a scene simply by raising or lowering the imaginary horizon—try a value of 10 to make the viewer feel he is crouching down close to the floor. Only if the vanishing point of the bottom grid is on the horizon would you feel as though you were standing normally on the floor.

The program draws the front-to-back lines first in Lines 4Ø to 8Ø and then the side-to-side lines in Lines 9Ø to 13Ø. The variable K gives the x coordinate of the start of each line and then is multiplied by V to give the x coordinate of the ends—along the top and bottom edge of the screen. The IF ... THEN condition in Line 5Ø just stops the computer drawing off the screen. This is not strictly necessary on the Acorns but it does make the program run faster.



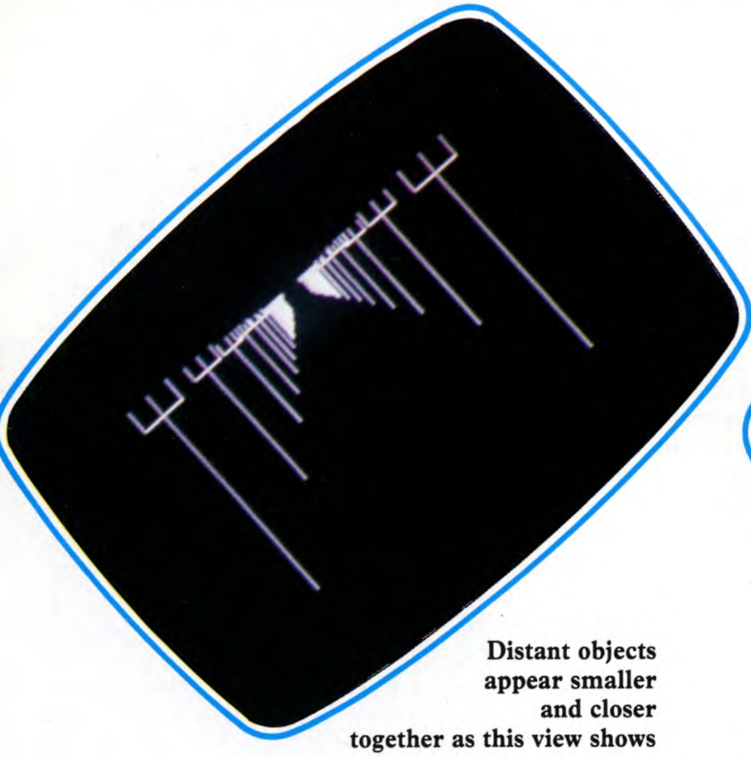
A perspective grid is a quick and easy way to add depth to your picture

The remaining lines are drawn by the next part of the program. The distance between them gets less and less as they get further into the distance and this is controlled by multiplying successive y coordinates by V↑1/6. The value of 1/6 was chosen to give the most realistic result, but you can experiment with different values.

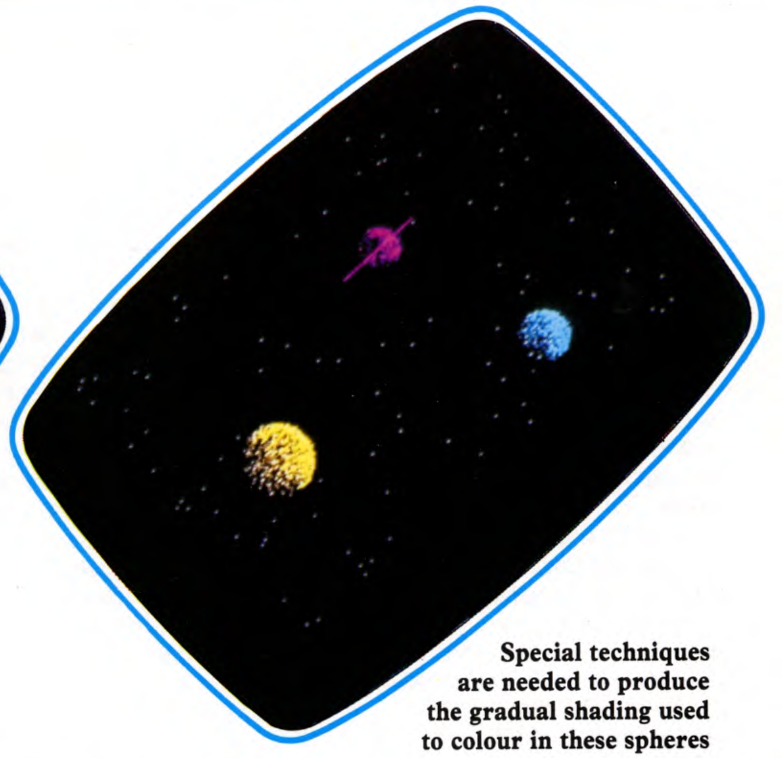
DIMINISHING SIZES

The rules of perspective apply to objects as well as lines on a grid. All objects appear smaller and closer together in the distance. The tops and bottoms of a line of equally-sized trees, for instance, lie on a pair of parallel lines which converge on the vanishing point.

The trick here is that the brain relates size to distance. If you see two objects which are known to be of similar size, but one appears to be half the size of the other, then the brain assumes that one is twice as far away. So by drawing objects diminishing in size, they can be made to appear to recede.



Distant objects appear smaller and closer together as this view shows



Special techniques are needed to produce the gradual shading used to colour in these spheres

If you think about the distances between objects, you will see that these, too, behave in the same way. Objects which are equally-spaced in reality will appear to get closer as the distance increases.

There is a strict mathematical relationship between these apparent differences—sizes appear according to the *inverse* of their distance from the observer. As the distance gets larger, 1/space gets smaller. That is why, in the first program, the separation between the horizontal lines was raised to the power of 1/6. The 6 here was simply chosen to give a suitable spacing, and changing it does not affect the appearance of recession.

The next program shows how to draw a perspective view of a roadway lined with telegraph poles:

```
10 BORDER Ø: PAPER Ø: INK 7: CLS
15 DEF FN Y(X)=((174-VP)/1ØØ)*(X-128)+VP
16 DEF FN B(X)=((2Ø-VP)/1ØØ)*(X-128)+VP
17 DEF FN S(T)=SF/SQR ((RW/2)↑2+(T*PH)↑2)
2Ø PRINT '''
3Ø INPUT "ENTER DISTANCE BETWEEN POLES□";P
4Ø INPUT "ENTER WIDTH OF ROAD□";RW
5Ø INPUT "ENTER HEIGHT OF POLES□";PH
6Ø INPUT "ENTER HEIGHT OF VIEW ABOVE ROAD□";RH
7Ø CLS
8Ø LET SF=1: LET SF=16Ø/FN S(Ø)
9Ø LET VP=16Ø/PH*RH+1ØØ: LET X=228: FOR T=1 TO 15
1ØØ LET X=X-1: IF FN S(T)<FN Y(X)-FN B(X) THEN GOTO 1ØØ
11Ø PLOT X,FN B(X): DRAW X-PEEK 23677,FN Y(X)-PEEK 23678: LET XJ=(FN Y(X)-FN B(X))/1Ø: LET YJ=FN Y(X)-XJ
12Ø PLOT X-XJ,FN Y(X): DRAW X-XJ-PEEK 23677,YJ-PEEK 23678: DRAW X+XJ-PEEK 23677,YJ-PEEK 23678: DRAW X+XJ-PEEK 23677,FN Y(X)-PEEK 23678
13Ø PLOT 255-X,FN B(X): DRAW 255-X-PEEK 23677,FN Y(X)-PEEK 23678
14Ø PLOT 255-X-XJ,FN Y(X):DRAW 255-X-XJ-PEEK 23677,YJ-PEEK 23678: DRAW 255-X+XJ-PEEK 23677,YJ-PEEK 23678: DRAW 255-X+XJ-PEEK 23677,FN Y(X)-PEEK 23678
15Ø NEXT T
16Ø GOTO 16Ø
```

```
1Ø DEFFNYT(X)=((9ØØ-VP)/5ØØ)*(X-64Ø)+VP:DEFFNYB(X)=((1ØØ-VP)/5ØØ)*(X-64Ø)+VP
2Ø DEFFNS(T)=SF/SQR((RW/2)↑2+(T*PH)↑2)
3Ø INPUT "♥DISTANCE BETWEEN POLES■";P
4Ø INPUT "WIDTH OF ROAD■";RW
5Ø INPUT "HEIGHT OF POLES■";PH
6Ø INPUT "HEIGHT OF VIEW ABOVE ROAD■";RH
7Ø HIRES Ø,1
8Ø SF=1:SF=8ØØ/FNS(Ø)
9Ø VP=8ØØ/PH*RH+1ØØ:X=114Ø:FOR T=1 TO 15
1ØØ X=X-4:IF FNS(T)<FNYT(X)-FNYB(X) THEN 1ØØ
11Ø LINE X/5,191-FNYB(X)/5,X/5,191-FNYT(X)/5,1
115 XJ=(FNYT(X)-FNYB(X))/1Ø:YJ=FNYT(X)-XJ
12Ø LINE (X-XJ)/5,191-FNYT(X)/5,(X-XJ)/5,191-YJ/5,1
125 LINE (X-XJ)/5,191-YJ/5,(X+XJ)/5,191-YJ/5,1
126 LINE (X+XJ)/5,191-YJ/5,(X+XJ)/5,191-FNYT(X)/5,1
13Ø LINE 255-X/5,191-FNYB(X)/5,255-X/5,191-FNYT(X)/5,1
14Ø LINE 255-(X+XJ)/5,191-FNYT(X)/5,255-(X+XJ)/5,191-YJ/5,1
145 LINE 255-(X+XJ)/5,191-YJ/5,255-(X-XJ)/5,191-YJ/5,1
146 LINE 255-(X-XJ)/5,191-YJ/5,255-(X-XJ)/5,191-FNYT(X)/5,1
15Ø NEXT T
16Ø GET A$:IF A$="" THEN 16Ø
17Ø NRM
18Ø RUN
```

```
1Ø MODE1:VDU19,Ø,7,Ø,Ø,Ø,19,3,Ø,Ø,Ø,Ø
2Ø PRINT'''
3Ø INPUT"DISTANCE BETWEEN POLES□",P
4Ø INPUT"WIDTH OF ROAD□",RW
5Ø INPUT"HEIGHT OF POLES□",PH
6Ø INPUT"HEIGHT OF VIEW ABOVE
   ROAD□",RH
7Ø CLS:VDU23;82Ø2;Ø;Ø;Ø;
8Ø SF=1:SF=8ØØ/FNS(Ø)
9Ø VP=8ØØ/PH*RH+1ØØ:X=114Ø:FOR
   T=1TO15
1ØØ REPEAT:X=X-4:UNTILFNS(T)>=
   FNYT(X)-FNYB(X)
11Ø MOVEX,FNYB(X):DRAWX,FNYT(X):XJ=
   (FNYT(X)-FNYB(X))/1Ø:YJ=FNYT(X)-
   XJ
12Ø MOVEX-XJ,FNYT(X):DRAWX-XJ,YJ:
   DRAWX+XJ,YJ:DRAWX+XJ,FNYT(X)
13Ø MOVE128Ø-X,FNYB(X):DRAW128Ø-X,
   FNYT(X)
14Ø MOVE128Ø-X-XJ,FNYT(X):DRAW
   128Ø-X-XJ,YJ:DRAW128Ø-X+XJ,YJ:
   DRAW128Ø-X+XJ,FNYT(X)
15Ø NEXT
16Ø D=GET
17Ø DEFFNYT(X)=((9ØØ-VP)/5ØØ)*(X-
   64Ø)+VP
18Ø DEFFNYB(X)=((1ØØ-VP)/5ØØ)*(X-
   64Ø)+VP
19Ø DEFFNS(T)=SF/SQR
   ((RW/2)^2+(T*PH)^2)
```

```
1Ø PMODE4,1:PCLS:CLS
2Ø DEFFNYT(X)=((9ØØ-VP)/5ØØ)*(X-
   64Ø)+VP:DEFFNYB(X)=((1ØØ-VP)/
   5ØØ)*(X-64Ø)+VP:DEFFNS(T)=SF/SQR
   ((RW/2)↑2+(T*PH)↑2)
3Ø INPUT"DISTANCE BETWEEN POLES ";P
4Ø INPUT"WIDTH OF ROAD ";RW
5Ø INPUT"HEIGHT OF POLES ";PH
6Ø INPUT"HEIGHT OF VIEW ABOVE ROAD
   ";RH
7Ø SCREEN1,1
8Ø SF=1:SF=8ØØ/FNS(Ø)
9Ø VP=8ØØ/PH*RH+1ØØ:X=114Ø:
   FORT=1TO15
1ØØ X=X-4:IFFNS(T)<FNYT(X)-FNYB
   (X) THEN1ØØ
11Ø LINE(X/5,191-FNYB(X)/5)-(X/5,
   191-FNYT(X)/5),PSET:XJ=(FNYT(X)-
   FNYB(X))/1Ø:YJ=FNYT(X)-XJ
12Ø LINE((X-XJ)/5,191-FNYT(X)/5)-
   ((X-XJ)/5,191-YJ/5),PSET:LINE-((X+
   XJ)/5,191-YJ/5),PSET:LINE-((X+
   XJ)/5,191-FNYT(X)/5),PSET
13Ø LINE(255-X/5,191-FNYB(X)/5)-
   (255-X/5,191-FNYT(X)/5),PSET
14Ø LINE(255-(X+XJ)/5,191-FNYT
   (X)/5)-(255-(X+XJ)/5,191-YJ/5),
   PSET:LINE-(255-(X-XJ)/5,191-
   YJ/5),PSET:LINE-(255-(X-XJ)/5,
   191-FNYT(X)/5),PSET
15Ø NEXT
16Ø IFINKEY$="" THEN16Ø ELSERUN
```

The programs let you specify exactly the view of the road and then draws an accurate perspective picture.

The program uses three functions to help the calculations. FNS(T) works out the actual height of each pole in pixels. FNYT(X) and FNYB(X) work out the y coordinates of the top and bottom of a pole for any x position. You can think of these two functions as drawing invisible lines along the tops and bottoms, converging together at the vanishing point.

In the main part of the program, Line 8Ø calculates a scale factor, SF, which uses the height of the previous pole to work out the height of the next one. The variable VP works out the y coordinate of the vanishing point, which depends on both your height above ground and the height of the poles. The x coordinate is always in the centre for this program.

Fifteen telegraph poles are drawn on each side of the road controlled by the loop in Lines 9Ø and 15Ø. The poles on the right-hand side are drawn first.

First of all, in Line 1ØØ, starting at the far right of the screen, the program decreases the x position until the height of the pole as calculated by FNS(T) fits exactly between the top and bottom lines worked out by FNYT and FNYB. When this happens, Line 11Ø draws the pole then the rest of the line works out XJ and YJ which are scale factors for the top part. Line 12Ø then draws the top part and the next two lines repeat the whole procedure for the poles on the left.

SHADING

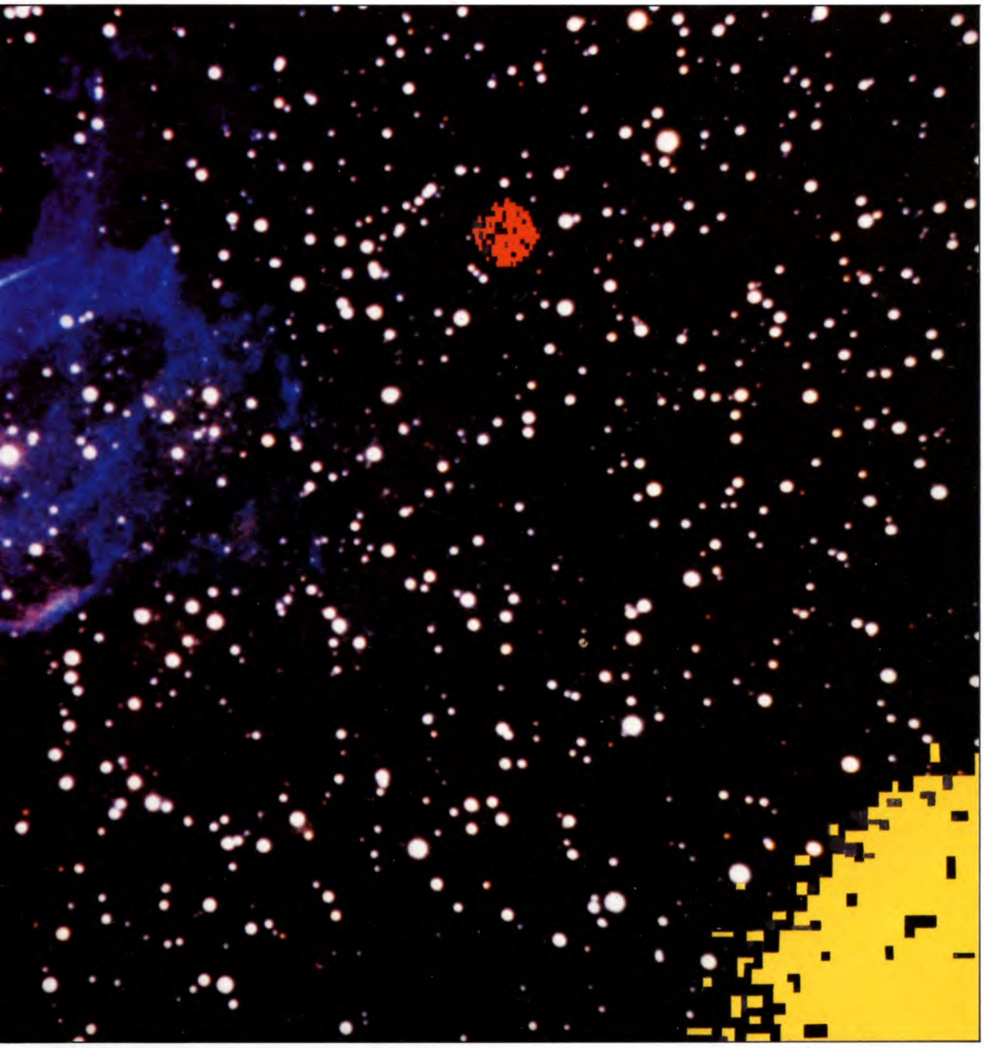
As well as perspective there are other effects you can use to give a feeling of depth and three-dimensionality to your graphics. The most useful of these is shading. Unfortunately, one of the great limitations of colour graphics on home computers is that there is no way of controlling the intensity of the colours. This makes it very difficult to create any realistic shading effects needed for solid-looking 3-D objects.

Large computers offer hundreds of shades of each colour so it is easy to draw realistic-looking objects. Have a look at the picture on page 421, for example. But home computers usually have no more than eight colours, so what's needed is a method of merging the colours more gradually.

PIXEL PLANETS

The way to add shading is to colour the pixels individually, lighting up only a few for dark areas and lighting almost all of them for bright areas. The number of coloured pixels in any area determines the brightness of that part of the object. So by gradually turning on more and more pixels from one side to the other you can create quite reasonable shading effects. The next program uses this technique to draw a group of spheres. They look so much like planets suspended in space that the program also adds a starry background and even a ring round one or two of the spheres!

```
1Ø BRIGHT Ø: BORDER Ø: PAPER Ø: INK 7:
   CLS
2Ø FOR I=1 TO 1ØØ: PLOT
   RND*255,RND*175: NEXT I
25 LET F=Ø: FOR T=1 TO 3
3Ø LET CL=RND*5+2
4Ø LET XC=RND*195+3Ø: LET
```



```
    YC=RND*115+3Ø
5Ø LET S=RND*3Ø
6Ø FOR K=-S TO S
7Ø IF INT (K)=Ø AND F=Ø THEN PLOT
    INVERSE 1; OVER 1;XC-L*2,YC: DRAW
    INK CL;L*4,Ø: LET F=1
8Ø LET X=SQR (S*S-K*K)
9Ø LET X2=2*X
1ØØ FOR L=-X TO X
11Ø PLOT INK CL; INVERSE 1;XC+L,YC+K:
    IF RND*X2-X<L THEN PLOT INK
    CL;XC+L,YC+K
12Ø NEXT L
13Ø NEXT K
14Ø NEXT T
15Ø PAUSE Ø
```

```
1Ø DEFFNA(X)=INT(RND(1)*X+1):HIRES 7,Ø
2Ø FOR K=1 TO 1ØØ:PLOT FNA(36Ø)-1,
    FNA(2ØØ)-1,1:NEXT K
25 FOR T=1 TO 8
3Ø XC=FNA(2ØØ)+3Ø:YC=FNA(13Ø)+3Ø
4Ø CL=FNA(3):CL=CL-(CL=3)
5Ø S=FNA(25)+5
6Ø FORK=-S TO S
7Ø IF K=1 AND FNA(4)=1 THEN CIRCLE
    XC,YC,S*1.5,FNA(4),1
8Ø X=SQR(S*S-K*K)
9Ø X2=2*X
1ØØ FOR L=-X TO X STEP 2
12Ø PLOT XC+L,YC-K,1
13Ø NEXT L,K
14Ø NEXT T
15Ø GOTO 15Ø
```

```
1Ø MODE1:VDU19,Ø,4,Ø,Ø,Ø,19,3,2,Ø,Ø,Ø
2Ø FORK=1TO2ØØ:PLOT69,RND(128Ø),RND
    (1Ø24):NEXT
25 FOR T=1 TO 8
3Ø VDU29,(RND(25Ø)+3Ø)*4;(RND
    (2ØØ)+3Ø)*4;
4Ø CL=RND(3):BC=Ø
5Ø S=RND(2Ø)+1Ø
6Ø FORK=-S□TO S
8Ø X=SQR(S*S-K*K)
9Ø X2=2*X
1ØØ FORL=-X TO□X
11Ø IF RND(X2)-X>L□THEN GCOLØ,CL□
    ELSE GCOLØ,BC
12Ø PLOT69,L*4,K*4
13Ø NEXT:NEXT
135 IF RND(2)=1 THEN MOVES*8,Ø:
    DRAW-S*8,Ø
14Ø NEXT
```

```
1Ø PMODE3,1:PCLS3:SCREEN1,Ø
2ØFORK=1TO1ØØ:PSET(RND(256)-1,RND
    (192)-1,2):NEXT
25 FOR T=1 TO 8
3Ø XC=RND(195)+3Ø:YC=RND(131)+3Ø
4Ø CL=RND(3):CL=CL-(CL=3)
5Ø S=RND(25)+5
6Ø FORK=-S TOS
7Ø IFK=1 ANDRND(4)=1 THENCIRCLE(XC,
    YC),S*1.5,RND(4),Ø
8Ø X=SQR(S*S-K*K)
9Ø X2=2*X
1ØØ FORL=-X TOX STEP2
11Ø IFRND(X2)-X<L THENCOLOR
    CL ELSECOLOR3
12Ø PSET(XC+L,YC-K)
13Ø NEXTL,K
14Ø NEXT T
15Ø GOTO 15Ø
```

The starry background is drawn by Line 2Ø which prints a hundred or so dots in random positions on the screen. Eight spheres are drawn in all (or three on the Spectrum) controlled by the loop in Lines 25 and 14Ø. Line 3Ø chooses a random position for the centre of each sphere, the next lines choose a random colour and a random size.

The general procedure for drawing each sphere is to start at the bottom and fill it in a line at a time. This is controlled by the value K which starts at −S, the y coordinate of the bottom of the circle and goes to +S at the top. Line 8Ø works out the x coordinate of the start of each line using the equation for a circle. X2 is the length of each line.

Lines 1ØØ to 13Ø colour in the lines. The variable L can be thought of as the luminosity or brightness of each region and its value depends on the distance from the edge of the circle, increasing from one side to the other. A random number, depending on the length of the line, is chosen by Line 11Ø, and if this is less than that region's brightness the colour is set to black. If it is more, then the colour is set to the random colour chosen earlier.

Finally, Line 7Ø on the Spectrum, Dragon and Tandy and Line 135 on the Acorns draw a 'ring' around one or two of the spheres.

See if you can adapt the routine to shade other shapes such as a cylinder, or a flat plane.

IN SEARCH OF THE BEST TIMES

Once you've told your computer about your project you can use the program to work out a realistic schedule and pinpoint any potential holdups

Before you can use the program to evaluate a particular project you have to break down the project into a number of individual activities and estimate the time taken for each one. To do this it is almost always best to sketch out a rough PERT network as shown in the example last time. It doesn't matter if the network gets rather tangled at this point as the computer will sort all that out for you.

As you draw out the chart, write the descriptions of the activities along the lines and a description of the events in the circles. Often, though, events won't need a description so you can leave them blank. The Spectrum will allow up to 20 characters for the activities and events, the Commodore allows up to 80 and the others up to 255. But to save on memory it's best to be brief. If you know more or less how long the activities will take then write the duration in too (see later for how to estimate the times). Remember, though, that all times must be in the same units whether they are hours, half-days, weeks or whatever.

In order for the program to work, the network must also be logically possible. There must be only one start and one finish point and there mustn't be any loops.

To help you plan out the chart keep asking yourself these three questions: 'what can be done at the same time as this activity?', 'what must be done before this activity can start?' and 'what cannot be started until this activity is completed?'.

Working out a PERT network forces you to think quite hard about what needs to be done. But the advantages are that you can then use the computer to work out the far more complicated questions of exactly when you should start all these activities, whether the job can be done at all, which activities are holding the whole project up, or which can you delay for a few hours, days or weeks.

INPUTTING ACTIVITIES

When you think you've worked out most of the chart the next thing to do is to number all the events and activities, and INPUT them into the program. The order of the numbers is not important but the computer needs them to work with. A common method is to number

events 10, 20, 30 and so on like line numbers, so that any extra ones inserted later can have intermediate numbers 15, 25 ... Activities can be numbered, for convenience, by referring to the start and end events. For instance 1020 is the activity between events 10 and 20. Or you can use the same numbers for both the events and activities if you like.

When defining the activities and events the computer prompts you for the number and description, and then it asks you for the average time and the 90% sure time. These are explained next.

INPUTTING THE TIMES

In real life you're very rarely going to be certain how long an activity will take—even if you've done it many times before. However, you can usually estimate the *average* time and take a guess at the *90% sure* time. This is the time within which you're fairly certain it will be done. And this is all the program requires you to do. These inputs are deceptively easy but in fact the program has to cover four quite different situations. You don't need to know how these are worked out as the program does it automatically, but it does help to understand what they are.

The first case is the rare occasion when you are absolutely certain of the time. For instance, if the instructions say 'leave for 24 hours' then that is what you must do. When using the program put both the average and the 90% sure times in as the same value.

The second case is the time that you are fairly sure about. For instance, you know you can drive to the station in about 30 minutes because you've done it many times before, but you allow 40 minutes to be on the safe side. For this you'd input 30 as the average time and 40 as the 90% sure time. This corresponds to the top graph in the diagram on page 1469, which is called a *normal Gaussian* curve.

The third type is the 'wait until it happens' time. For instance, you won't know if the roof repairs have worked until it rains. This is shown in the second graph. Here the 90% sure time is about two and a half times the average—the average in this case being found from records of rainy days for the month.

The fourth case is the 'all or nothing' time. For instance, it may be very unlikely that a crucial part in a car is broken (say one in a hundred) but if it is it will take 10 days to repair. In this case you input the maximum time (10 days) as the 90% sure time, and the arithmetic average (10 times $\frac{1}{100}$, or $\frac{1}{10}$ of a day) as the average. This is shown in the third graph.

There is no need to tell the computer which type of graph you're using (if any). The computer just takes your two time estimates and proceeds accordingly. If they are approximately equal (up to a ratio of 1 to $1\frac{1}{3}$) it uses the Gaussian curve (top graph). If they are further apart (the 90% sure time between $1\frac{1}{3}$ and $2\frac{1}{3}$ times the average) it uses a modified Gaussian curve (not shown). If it is between $2\frac{1}{3}$ and 3 times the average it uses the exponential curve (middle graph). And if it is more than three times the average it uses the bimodal curve (bottom graph).

The reason the program needs to take so much care over the uncertain times is because the critical path may very well change if all or some of the uncertainties conspire to their worst cases (or their best).

INPUTTING THE EVENTS

When you've input all the activities you should then enter the events. This is very easy, simply enter a number and a description for every event on your chart.

If you find you have made a mistake you can delete any event or activity with the delete option or alter it by defining it again and entering the correct values.

The information entered into the computer can be displayed in several different tables. So, assuming you've typed in the activities and events, choose the option to Show Details and you'll see a neat list of everything you've entered. If you have a printer connected you can get a print-out too.

The point of the program, though, is to calculate the critical path through your project so you can work out the most efficient way to carry out all of your activities.

DATA CHECK

Before the computer can make any calculations it must check that the network is logical. If there were any loops the program would go round in circles trying to do the calculations and the program would crash. The Acorn does the data check automatically when you ask for a calculation but with the other programs you have to choose this option yourself. If all is well, the program will print out the numbers of the start event and the end event. If there are any inconsistencies the program will print out a message telling you exactly what's wrong, identifying any loops or breaks.

THE CRITICAL PATH

At last you can select the option to calculate the critical path. There are two options. The first uses the average times that you input for the activities while the second uses the uncertain times. Run the average time first.

The display shows each activity with its code number and description. It then tells you the time when this activity is able to start, the time it must finish, if there is any slack, and whether this particular activity is actually critical.

The times are in the same units as those you input for the activities. So if you've used days, all the figures in the display refer to days. For example, if the display tells you that activity 3 is able to start 6, must finish 10 and has slack 2, this means the earliest you can start is on day six but with two days slack you could, if it is more convenient, put off starting until day eight without upsetting the whole project.

The activities that have slack 0 mean that you *must* start on the day shown or the project will be delayed. These are the critical activities and it's a good idea to mark these on your chart, in red, say. If the starting date for these activities starts to slip you'll have to think about rearranging the rest of your project to make up the time.

One of the advantages of this program is that you can try out many different arrangements of activities to find the quickest, or most efficient.

UNCERTAIN TIMES

If many of the times you input were uncertain, and the 90% sure time was different to the average time, then you should use the other calculate option which takes these uncertainties into account. The critical path might change when uncertain times are used.

When you choose this option the computer takes each activity and, using the appropriate graph mentioned earlier, it selects a random time within the limits allowed. Using these times it then calculates the critical path for the whole network exactly as it did in the last option. It also stores away in memory the start, end and slack times for each activity. It does this 44 more times, selecting a new random number each time. (The computer's progress is shown on the screen.) The 45 cases are needed to give a reasonably random sample. The final display printed out takes all these samples into account.

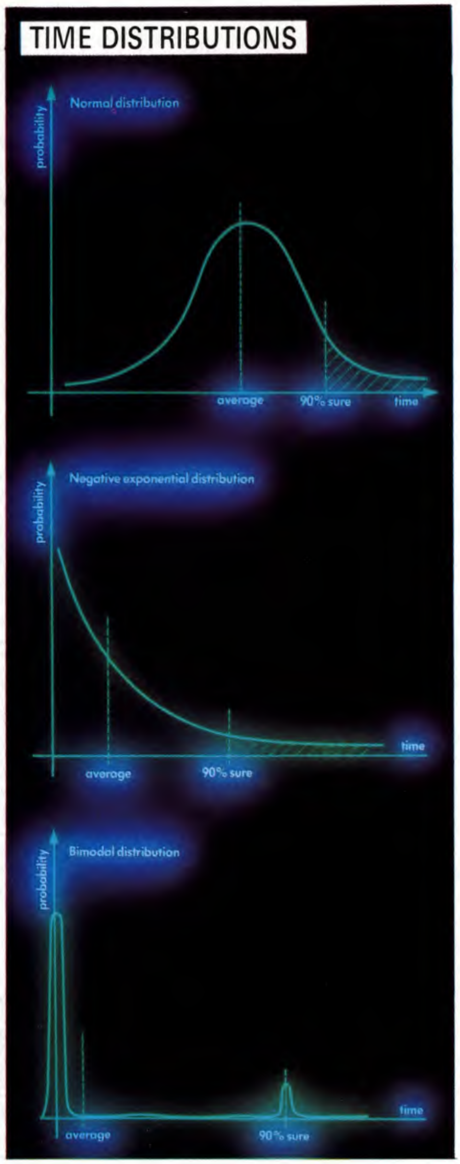
The start and end times are averages of the 45 cases and so are quite reliable. The slack time is also an average, but only of those times when there was some slack—a critical activity gives no slack at all.

The critical value shows the percentage of times that the activity was part of the critical path. This may be 100% in which case it is always critical, 0% when it is never critical, or any value in between. For instance, an activity may be critical 33% of the time which means that the probability of it being critical on any one occasion is a third.

Finally, the last value that's shown is the standard deviation of the slack time. This tells you how much the slack time is likely to vary and gives you an idea of how reliable it is. For example, if the slack was 1.5 and the deviation was 1 then the slack may vary from about .5 to 2.5, so the slack time printed out cannot be relied on. If, however, the slack was 1.5 and the deviation was .1 then there's likely to be little variation, so the slack time is reliable.

```
330 PRINT w$(4);f$;":";: INPUT u$(x):
    PRINT u$(x): LET s(x)=Ø: RETURN
350 LET ee=ee+1: LET e(ee)=x: LET
    s(x)=-1: LET f(x)=Ø: LET u(x)=u
360 LET t(x)=Ø: LET n(x)=Ø: LET
    u$(x)="": RETURN
400 LET z=x: FOR f=1 TO ee: IF e(f)=z
    THEN LET e=f
420 NEXT f: LET e(e)=e(ee): LET
    u(z)=zz+1: LET ee=ee-1: RETURN
450 LET z=u-INT ((u-1)/mh)*mh: LET
    y=2: LET x=Ø
460 IF x=Ø THEN IF Ø=u(z) OR
    zz+1=u(z) THEN LET x=z
470 IF u=u(z) THEN LET x=z: RETURN
480 IF y=1 OR□Ø=u(x) THEN RETURN
490 LET z=z+y-mh*INT ((z+y-1)/mh):
    LET y=y+y-mh*INT ((y+y-1)/mh):
    GOTO 460
500 LET x(1)=ma: LET x(2)=me: LET
    x(3)=mh: LET x(4)=aa: LET x(5)=ee:
    LET x(6)=ck: LET x(7)=se: LET x(8)=fe:
    PRINT "press[ENTER] ten times": SAVE
    f$+"x" DATA x(): FOR x=1 TO 100:
    NEXT x
510 SAVE f$+"a" DATA a(): SAVE f$+"e"
    DATA e(): SAVE f$+"f" DATA f(): SAVE
    f$+"g" DATA g(): SAVE f$+"n" DATA
    n(): SAVE f$+"s" DATA s(): SAVE
    f$+"t" DATA t(): SAVE f$+"u" DATA
    u(): SAVE f$+"u$" DATA u$(): RETURN
600 LOAD f$+"x" DATA x(): LET ma=x(1):
    LET me=x(2): LET mh=x(3): LET
    aa=x(4): LET ee=x(5): LET ck=x(6): LET
    se=x(7): LET fe=x(8): GOSUB 12
610 LOAD f$+"a" DATA a(): LOAD f$+"e"
    DATA e(): LOAD f$+"f" DATA f(): LOAD
    f$+"g" DATA g(): LOAD f$+"n" DATA
    n(): LOAD f$+"s" DATA s(): LOAD
    f$+"t" DATA t(): LOAD f$+"u" DATA
    u(): LOAD f$+"u$" DATA u$(): LET
    false=Ø: RETURN
700 LET x(1)=ma: LET x(2)=me: LET
    x(3)=mh: LET x(4)=aa: LET x(5)=ee:
    LET x(6)=ck: LET x(7)=se: LET x(8)=fe:
    VERIFY f$+"x" DATA x()
710 VERIFY f$+"a" DATA a(): VERIFY
    f$+"e" DATA e(): VERIFY f$+"f" DATA
    f(): VERIFY f$+"g" DATA g(): VERIFY
    f$+"n" DATA n(): VERIFY f$+"s" DATA
    s(): VERIFY f$+"t" DATA t(): VERIFY
    f$+"u" DATA u(): VERIFY f$+"u$"
    DATA u$(): RETURN
800 GOSUB 942
810 FOR a=1 TO aa: LET x=a(a): GOSUB
    932
820 LET y=y+1+ (LEN u$(x)>4): IF
    y>20 AND a<aa THEN GOSUB 940:
    GOSUB 942
830 NEXT a: GOSUB 940
840 GOSUB 946: FOR e=1 TO ee: LET
    x=e(e): GOSUB 933
850 LET y=y+1+(LEN u$(x)>4): IF
    y>20 AND e<ee THEN GOSUB 940:
    GOSUB 946
860 NEXT e: GOTO 940
932 PRINT FN I$(FN u(s(x)));FN I$(FN
    u(f(x)));FN I$(t(x));FN I$(n(x));ABS u(x);
    "□";u$(x): RETURN
933 PRINT ABS u(x) ,u$(x): RETURN
940 PRINT "press [ENTER] to continue":
    INPUT f$: CLS : RETURN
942 CLS : PRINT "START FINISH TIME 90%
    CODE TEXT"
944 PRINT "EVENT EVENT ALLOW SURE":
    LET y=3: RETURN
946 PRINT "CODE","TEXT": RETURN
948 PRINT "PREV AFTER MIN□□MAX":
    LET y=3: RETURN
1000 LET ck=true: FOR a=1 TO aa: LET
    x=a(a)
1020 LET z=s(x): IF s(z)<Ø OR zz<u(z)
    THEN PRINT u(x);w$(5);u(z): LET
    ck=false
1030 LET z=f(x): IF s(z)<Ø OR zz<u(z)
    THEN PRINT u(x);w$(5);u(z): LET
    ck=false
1040 NEXT a: IF ck=false THEN GOTO 1750
1050 LET e=1
1060 LET z=e(e): IF s(z)<Ø THEN GOSUB
    400: IF e<=ee THEN GOTO 1060
1070 LET e=e+1: IF e<=ee THEN GOTO
    1060
1080 FOR e=1 TO ee: LET z=e(e): LET
    s(z)=Ø: LET f(z)=Ø: NEXT e
1082 FOR a=1 TO aa: LET x=a(a): LET
    s(f(x))=x: NEXT a
1090 LET se=Ø: FOR e=1 TO ee: LET
    z=e(e): IF s(z)>Ø THEN GOTO 1096
1092 IF se=Ø THEN LET se=z: GOTO 1096
1094 PRINT w$(1);u(z): IF se<=mh THEN
    PRINT w$(1);u(se): LET se=mh+1
1096 NEXT e: IF se=Ø THEN PRINT "ALL
    EVENTS HAVE PRECEDING";a$
1098 IF se=Ø OR se>mh THEN GOTO 1750
1100 FOR e=1 TO ee: LET z=e(e): LET
    t(z)=Ø: LET n(z)=Ø: NEXT e: LET
    t(se)=1
1110 LET last=1: FOR c=2 TO ee+2: IF
    last<>c-1 THEN GOTO 1170
1120 FOR a=1 TO aa: LET x=a(a): LET
    y=s(x): IF t(y)<>c-1 THEN GOTO
    1160
1130 IF y=f(x) THEN GOSUB 1200: GOTO
    1160
1140 IF y<>se THEN LET y=s(y): GOTO
    1130
1150 LET y=f(x): LET s(y)=s(x): LET
    f(s(y))=y: LET t(y)=c: LET fe=y: LET
    last=c
1160 NEXT a
1170 NEXT c: PRINT "start event=";u(se);",
    end event=";u(fe)
1180 FOR e=1 TO ee: LET y=e(e)
1190 IF f(y)=Ø AND y<>fe THEN PRINT
    u(y);"NOT LINKED TO END EVENT": LET
    ck=false
1192 NEXT e: IF ck THEN GOTO 1300
1194 GOTO 1750
1200 CLS : PRINT "THERE IS A LOOP AS
    FOLLOWS": PRINT "EVENTS ...": LET
    xa=a(a)
1210 LET x=f(xa): PRINT u(x): LET y=s(xa):
    PRINT u(y)
1220 LET y=s(y): PRINT u(y): IF y<>x
    THEN GOTO 1220
1230 RETURN
1300 LET k=1: LET ak=aa: IF aa=1 THEN
    LET k=Ø
```

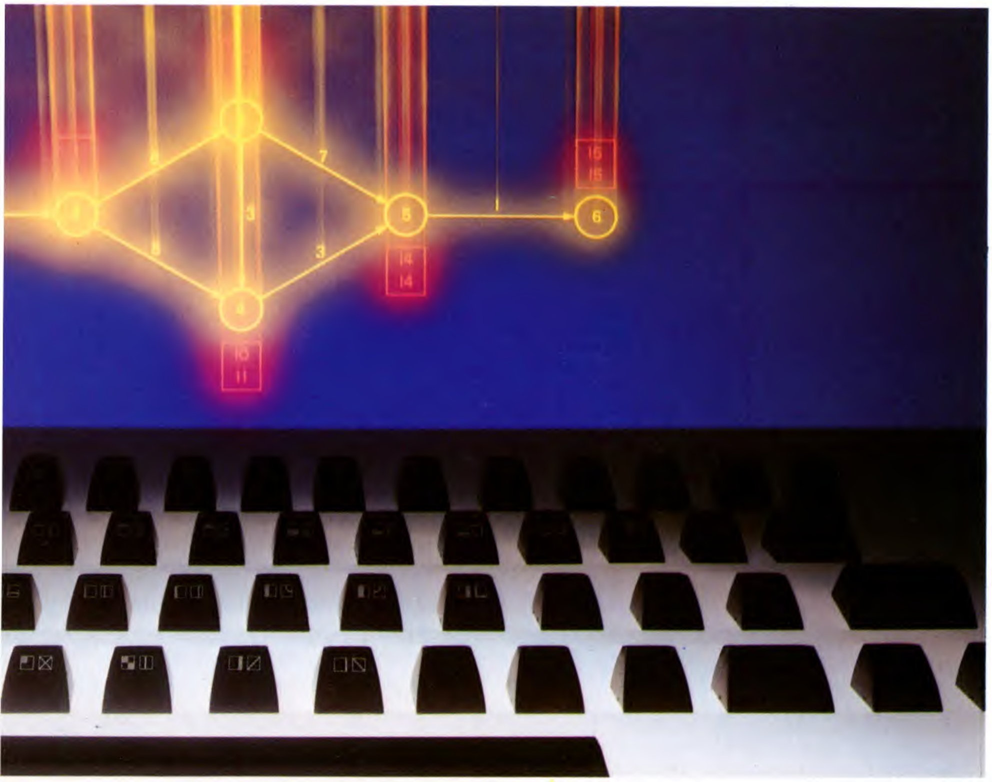
```
1310 LET ak=INT ((ak+k)/2): IF ak=Ø
   THEN GOTO 15ØØ
1320 LET k=Ø: FOR a=ak+1 TO aa: LET
   b=a-ak: LET x=a(a): LET y=a(b): LET
   xe=s(x): LET ye=s(y)
1330 IF t(ye)+ye/zz<=t(xe)+xe/zz THEN
   GOTO 136Ø
1340 LET a(a)=y: LET a(b)=x: LET k=1
1360 NEXT a: GOTO 131Ø
15ØØ LET n(fe)=last: FOR d=last-1 TO 1
   STEP -1
152Ø FOR a=1 TO aa: LET x=a(a): IF
   n(f(x))<>d+1 THEN GOTO 156Ø
155Ø LET y=s(x): LET f(y)=f(x): LET
   n(y)=d
156Ø NEXT a: NEXT d
16ØØ FOR a=1 TO aa: LET g(a)=a(a): NEXT
   a: LET k=1 LET ak=aa: IF aa=1 THEN
   LET k=Ø
161Ø LET ak=INT ((ak+k)/2): IF ak=Ø
   THEN GOTO 17ØØ
162Ø LET k=Ø: FOR a=ak+1 TO aa: LET
   b=a-ak: LET x=g(a): LET y=g(b): LET
   xe=f(x): LET ye=f(y)
163Ø IF n(ye)+ye/zz<=n(xe)+xe/zz THEN
   GOTO 166Ø
164Ø LET g(a)=y: LET g(b)=x: LET k=1
166Ø NEXT a: GOTO 161Ø
17ØØ LET ck=true: RETURN
175Ø LET ck=false: PRINT AT 21,8;"ANY KEY
   TO CONTINUE": PAUSE Ø: RETURN
2ØØØ FOR a=1 TO aa: LET x=a(a): LET
   z(x)=t(x): NEXT a: GOSUB 21ØØ
2Ø2Ø FOR a=1 TO aa: LET x=a(a): LET y(x)
   =(z(f(x))-y(s(x))=z(x))*1ØØ: NEXT a
2Ø3Ø FOR b=1 TO aa STEP 5: CLS : FOR
   a=b TO aa+FN a(b+4-aa): LET
   x=a(a)
2Ø4Ø PRINT a$;"□";u(x); "=";u$(x)(TO16)
2Ø5Ø LET c=y(s(x)): LET d=z(f(x)): PRINT
   "can start□";c;",must end□";d
2Ø6Ø PRINT "slack□";d-c-z(x);"
   □(critical□";y(x);"%)": IF t=12 THEN
   PRINT "std devn=";q(x);
2Ø7Ø PRINT : NEXT a: GOSUB 94Ø: NEXT b:
   RETURN
21ØØ FOR e=1 TO ee: LET y(e(e))=Ø: NEXT
   e
211Ø FOR a=1 To aa: LET x=a(a): LET y(f
   (x))=y(f(x))+FN z(y(s(x))-y(f(x))
   +z(x)):NEXT a
212Ø FOR e=1 TO ee: LET z(e(e))=y(fe):
   NEXT e: FOR a=aa TO 1 STEP -1: LET
   x=g(a)
213Ø LET z(s(x))=z(s(x))+FN a(z(f(x))
   -z(s(x))-z(x)): NEXT a: RETURN
3ØØØ FOR a=1 TO aa: LET x=a(a): LET
   p(x)=Ø: LET q(x)=Ø: LET y(x)=Ø: NEXTa
3Ø2Ø FOR e=1 TO ee: LET z=e(e): LET
   p(z)=Ø: LET q(z)=Ø: NEXT e
3Ø3Ø FOR m=1 TO 43 STEP 3: FOR a=1 TO
   aa: LET w(a)=2*RND-1: NEXT a
3Ø4Ø FOR n=Ø TO 4 STEP 2: CLS : PRINT
   "STARTING CASE□";m+n/2;"□OF 45"
3Ø5Ø FOR a=1 TO aa: LET x=a(a): LET
   tx=t(x): IF tx=Ø THEN LET z(x)=Ø:
   GOTO 3Ø8Ø
3Ø52 LET nx=n(x): IF nx=tx THEN LET
   z(x)=tx: GOTO 3Ø8Ø
3Ø54 LET w=FN w(w(a)+n/3): IF
   nx>=tx*3 THEN LET
   z(x)=nx*(w<tx/nx): GOTO 3Ø8Ø
3Ø6Ø IF nx>tx*2.34 THEN LET z(x)=-tx*LN
   w: GOTO 3Ø8Ø
3Ø7Ø LET w=FN x(w-.5): LET z(x)=ABS
   (tx+w*(nx-tx))
3Ø8Ø NEXT a
3Ø9Ø GOSUB 21ØØ
31ØØ FOR a=1 TO aa: LET x=a(a): LET z=z
   (f(x))-y(s(x))-z(x)
311Ø LET p(x)=p(x)+z: LET q(x)=q(x)
   +z*z: LET y(x)=y(x)+(z<1.e-6):
   NEXT a
312Ø FOR e=1 TO ee: LET z=e(e): LET
   p(z)=p(z)+y(z): LET q(z)=q(z)+z(z):
   NEXT e: NEXT n: NEXT m
32ØØ FOR e=1 TO ee: LET z=e(e): LET
   y(z)=VAL (FN I$(p(z)/45))
321Ø LET z(z)=VAL (FN I$(q(z)/45)): NEXT e
322Ø FOR a=1 TO aa: LET x=a(a): LET
   y=y(x): LET y(x)=VAL ((STR$
   (y/45*1ØØ)+"□□□")( TO 4))
323Ø IF p(x)<1.e-2 THEN LET p(x)=Ø
324Ø LET z=(45-y)+.1e-9: LET z(x)
   =z(f(x))-y(s(x))-VAL (FN I$(p(x)/Z))
325Ø LET q(x)=SQR ABS ((q(x)-p(x)*
   p(x)/z)/((z-1)+.1e-9)): IF
   q(x)<1.e-6 THEN LET q(x)=Ø
326Ø NEXT a: GOTO 2Ø3Ø
```




```
6ØØ OPEN1,8,8,"Ø:"+F$+",S,R":INPUT
   #1,MA,ME,MH,AA,EE,CK:GOSUB12
61Ø IFCKTHENINPUT#1,SE,FE
62Ø FORA=1TOAA:INPUT#1,X,U%(X),
   S%(X),F%(X),T(X),N(X),G%(A),U$(X):
   A%(A)=X:NEXTA
64Ø FORE=1TOEE:INPUT#1,X,U%(X),
   S%(X),F%(X),T(X),N(X),U$(X):E%(E)=
   X:NEXTE
65Ø INPUT#1,X:IFX>ØTHEN
   U%(X)=ZZ+1:GOTO65Ø
66Ø CLOSE1:RETURN
7ØØ OPEN15,8,15,"SØ:"+F$:CLOSE15:
   RETURN
8ØØ GOSUB942
81Ø FORA=1TOAA:X=A%(A):GOSUB932
82Ø Y=Y+1-(LEN(U$(X))>12):IFY>2Ø
   AND(A<AA)THENGOSUB94Ø:
   GOSUB942
83Ø NEXTA:GOSUB94Ø:PRINT"■EVENTS"
   :Y=3
84Ø FORE=1TOEE:X=E%(E):XP=U%(X):
   GOSUB95Ø:PRINTU$(X)
85Ø Y=Y+1-(LEN(U$(X))>12):IFY>2Ø
   AND E<EE THENGOSUB94Ø:GOSUB946
86Ø NEXTE:GOTO94Ø
9ØØ INPUT "■RESTART PROGRAM (Y/N)";
   AN$:IF AN$="N" THEN 5Ø
91Ø IF AN$<>"Y" THEN 9ØØ
92Ø RUN
932 XP=FNU(S%(X)):GOSUB95Ø:XP=FNU
   (F%(X)):GOSUB95Ø:XP=T(X):GOSUB95Ø:
   XP=N(X)
933 GOSUB95Ø:XP=ABS(U%(X)):
   GOSUB95Ø
935 PRINT:PRINT "TEXT=";U$(X):RETURN
94Ø IFKK$<>"Y" THENPRINT"■PRESS
   RETURN TO CONTINUE":INPUTF$:PRINT
   CLS$;:RETURN
941 RETURN
942 PRINTCLS$"■ACTIVITIES"
```

```
943 PRINT"START FINISH TIME□ □ 90%"
944 PRINT"EVENT EVENT□ □ALLOW SURE
    □ □CODE":Y=3:RETURN
950 XP$=LEFT$(STR$(XP)+"□□□□□
    □",6):PRINTXP$;:RETURN
960 PRINT "♥OUTPUT TO PRINTER (Y/N)?"
970 GET KK$:IF KK$<>"N" AND KK$<>
    "Y" THEN 970
980 RETURN
1000 CK=TR:FORA=1TOAA:X=A%(A)
1020 XE=S%(X):IFS%(XE)<ØORZZ<U%
    (XE)THENPRINTU%(X);W$(5)U%(XE):
    CK=FA
1030 Z=F%(X):IFS%(Z)<ØORZZ<U%(Z)
    THENPRINTU%(X);W$(5);U%(Z):CK=FA
1040 NEXTA:IFCK=FATHEN1750
1050 E=1
1060 X=E%(E):IFS%(X)<ØTHENGOSUB
    400:IF E<=EE THEN1060
1070 E=E+1:IFE<=EE THEN1060
1080 FORE=1TOEE:X=E%(E):S%(X)=Ø:F%
    (X)=Ø:NEXTE
1082 FORA=1TOAA:X=A%(A):S%(F%(X))
    =X:NEXTA
1090 SE=Ø:FORE=1TOEE:X=E%(E):IFS%
    (X)>ØTHEN1096
1092 IFSE=ØTHENSE=X:GOTO1096
1094 PRINTW$(1);U%(X):IFSE<=MHTHEN
    PRINTW$(1);U%(SE):SE=MH+1
1096 NEXTE:IFSE=ØTHENPRINT"ALL
    EVENTS HAVE PRECEEDING▮";A$
1098 IFSE=ØOR(SE>MH)THEN1750
1100 FORE=1TOEE:X=E%(E):T(X)=Ø:
    N(X)=Ø:NEXTE:T(SE)=1
1110 LA=1:FORC=2TOEE+2:IFLA
    <>C-1THEN1170
1120 FORA=1TOAA:X=A%(A):Y=S%(X):
    IFT(Y)<>C-1THEN1160
1130 IFY=F%(X)THENGOSUB1200:GOTO
    1160
1140 IF(Y<>SE)THENY=S%(Y):GOTO1130
1150 Y=F%(X):S%(Y)=S%(X):F%(S%(Y))=
    Y:T(Y)=C:FE=Y:LA=C
1160 NEXTA
1170 NEXTC:PRINT"START EVENT=";
    U%(SE);", END EVENT=";U%(FE)
1180 FORE=1TOEE:Y=E%(E)
1190 IFF%(Y)=ØAND(Y<>FE)THENPRINT
    U%(Y);"NOT LINKED TO END EVENT"
    :CK=FA
1192 NEXTE:IFCKTHEN1300
1194 GOTO1750
1200 PRINTCLS$;"THERE IS A LOOP AS
    FOLLOWS":PRINT"EVENTS...":XA=A%(A)
1210 X=F%(XA):PRINTU%(X):Y=S%(XA):
    PRINTU%(Y)
1220 Y=S%(Y):PRINTU%(Y):IFY<>XTHEN
    1220
1230 RETURN
1300 K=1:AK=AA:IFAA=1THENK=Ø
1310 AK=INT((AK+K)/2):IFAK=ØTHEN
    1500
1320 K=Ø:FORA=AK+1TOAA:B=A
    -AK:X=A%(A):Y=A%(B):XE=S%(X):
    YE=S%(Y)
1330 IFT(YE)+YE/ZZ<=T(XE)+XE/ZZ
    THEN1360
1340 A%(A)=Y:A%(B)=X:K=1
1360 NEXTA:GOTO1310
1500 N(FE)=LA:FORD=LA-1TO1STEP-1
1520 FORA=1TOAA:X=A%(A):IFN
    (F%(X))<>D+1THEN1560
1550 Y=S%(X):F%(Y)=F%(X):N(Y)=D
1560 NEXTA,D
1600 FORA=1TOAA:G%(A)=A%(A):
    NEXTA:K=1:AK=AA:IFAA=1THENK=Ø
1610 AK=INT((AK+K)/2):IFAK=ØTHEN
    1700
1620 K=Ø:FORA=AK+1TOAA:B=A-AK:
    X=G%(A):Y=G%(B):XE=F%(X):
    YE=F%(Y)
1630 IFN(YE)+YE/ZZ<=N(XE)+XE/ZZ
    THEN1660
1650 G%(B)=X:G%(A)=Y:K=1
1660 NEXTA:GOTO1610
1700 CK=TR:RETURN
1750 CK=FA:INPUT"HIT RETURN";HG$:
    RETURN
2000 FORA=1TOAA:X=A%(A):Z(X)=T(X):
    NEXTA:GOSUB2100
2020 FORA=1TOAA:X=A%(A):Y(X)=
    -(Z(F%(X))-Y(S%(X))=Z(X))*100:
    NEXTA:GOSUB2100
2030 FORB=1TOAASTEP4:PRINTCLS$:FOR
    A=BTOAA+FNA(B+4-AA):X=A%(A)
2040 PRINT:PRINTA$;U%(X);"=";U$(X)
2050 C=Y(S%(X)):D=Z(F%(X))
2055 PRINT"ABLE TO START";INT
    (C)*100/100;"MUST FINISH";INT(D*100)
    /100
2060 PRINT"SLACK";INT((D-C-Z(X))*
    100)/100;"(CRITICAL";INT(Y(X)*100)/
    100;"%)"
2065 IFT=12THENPRINT"STD DEVN="INT
    (Q(X)*100)/100;
2070 PRINT:NEXTA:GOSUB940:NEXTB:
    RETURN
2100 FORE=1TOEE:Y(E%(E))=Ø:NEXTE
2110 FORA=1TOAA:X=A%(A):Y(F%(X))=
    Y(F%(X))+FNZ(Y(S%(X))-Y(F%(X))+Z
    (X)):NEXTA
2120 FORE=1TOEE:Z(E%(E))=Y(FE):
    NEXTE:FORA=AA TO1STEP-1:X=G%(A)
2130 Z(S%(X))=Z(S%(X))+FNA(Z(F%(X))
    -Z(S%(X))-Z(X)):NEXTA:RETURN
3000 FORA=1TOAA:X=A%(A):P(X)=Ø:Q
    (X)=Ø:Y(X)=Ø:NEXTA
3020 FORE=1TOEE:X=E%(E):P(X)=Ø:Q
    (X)=Ø:NEXTE
3030 FORM=1TO43STEP3:FORA=1TOAA:
    W(A)=2*RND(Ø)-1:NEXTA
3040 FORN=ØTO4STEP2:PRINTCLS$;
```

```
    "STARTING CASE"M+N/2"OF 45"
3050 FORA=1TOAA:X=A%(A):TX=T(X):IF
    TX=ØTHENZ(X)=Ø:GOTO3080
3052 NX=N(X):IF(NX=TX)THENZ(X)=TX:
    GOTO3080
3054 W=FNW(W(A)+N/3):IFNX>=TX*3
    THENZ(X)=-NX*(W<TX/NX):GOTO 3080
3060 IFNX>TX*2.34THENZ(X)=-TX*LOG
    (W):GOTO3080
3070 W=FNX(W-.5):Z(X)=ABS(TX+W*
    (NX-TX))
3080 NEXTA
3090 GOSUB2100
3100 FORA=1TOAA:X=A%(A):Z=Z(F%(X))
    -Y(S%(X))-Z(X)
3110 P(X)=P(X)+Z:Q(X)=Q(X)+Z*Z:Y(X)
    =Y(X)+(Z<1.E-6):NEXTA
3120 FORE=1TOEE:X=E%(E):P(X)=P(X)
    +Y(X):Q(X)=Q(X)+Z(X):NEXTE,N,M
3125 IF KK$="Y" THEN OPEN 4,4:CMD4
3200 FORE=1TOEE:X=E%(E):Y(X)=VAL
    (LEFT$(STR$(P(X)/45),6))
3210 Z(X)=VAL(LEFT$(STR$(Q(X)/45),6)):
    NEXTE
3220 FORA=1TOAA:X=A%(A):Y=Y(X):
    Y(X)=-VAL(LEFT$(STR$(Y/45*100),4))
3230 IFP(X)<1.E-2THENP(X)=Ø
3240 Z=(45-Y)+.1E-9:Z(X)=Z(F%(X))
    -Y(S%(X))-VAL(LEFT$(STR$(P(X)/Z),6))
3250 Q(X)=SQR((Q(X)-P(X)*P(X)/Z)/((Z
    -1)+.1E-9)):IFQ(X)<1.E-6THENQ
    (X)=Ø
3260 NEXTA:GOTO2030
```



```
500 IFU%(X)<ØGOSUB87Ø:PRINT;ABS(U%
    (X))TAB(1Ø)U$(X):GOTO 53Ø
510 IF(EE=ME)THENPRINTW$(2);F$:
    RETURN
520 GOSUB54Ø
530 PRINTW$(4);F$:INPUTU$(X):
    S?X=Ø:RETURN
540 EE=EE+1:E?EE=X:S?X=-1:F?X=Ø:
    U%(X)=U
550 T(X)=Ø:U$(X)="":RETURN
560 Z=X:FORF%=1TOEE:IFZ=E?F%THEN
    E%=F%
570 NEXTF%:E?E%=E?EE:U%(Z)=ZZ+1:
    EE=EE-1:RETURN
580 Z=U-INT((U-1)/MH)*MH:Y=2:X=Ø
590 IFX=ØTHENIFØ=U%(Z)ORZZ+1=U%
    (Z)THENX=Z
600 IFU=U%(Z)THENX=Z:RETURN
610 IFY=1ORØ=U%(Z)THENRETURN
620 Z=Z+Y-MH*INT((Z+Y-1)/MH):
    Y=Y+Y-MH*INT((Y+Y-1)/MH):
    GOTO59Ø
630 IFAA=ØOREE=ØTHENRETURN
    ELSEO=OPENOUT(F$):PRINT#O,MA,
    ME,MH,AA,EE,CK
640 IF(CK)THENPRINT#O,SE,FE
650 FORA%=1TOAA:X=A?A%:PRINT#O,X,
    U%(X) ,S?X,F?X,T(X) ,N(X) ,G?A%,U$(X):
    NEXTA%
660 FORE%=1TOEE:X=E?E%:PRINT#O,X,
    U%(X) ,S?X,F?X,T(X) ,N(X) ,U$(X): NEXTE%
670 FORX=1TOMH:IFU%(X)=ZZ+1THEN
    PRINT#O,X
680 NEXTX:PRINT#O,Ø
690 CLOSE#O:RETURN
700 I=OPENIN(F$):INPUT#I,MA,ME,MH,
    AA,EE,CK:GOSUB2Ø
710 IF(CK)THENINPUT#I ,SE,FE
720 FORA%=1TOAA:INPUT#I,X,U%(X)
    ,S?X,F?X,T(X),N(X) ,G?A%,U$(X):
    A?A%=X:NEXTA%
730 FORE%=1TOEE:INPUT#I,X,U%(X) ,S?X,
    F?X,T(X) ,N(X) ,U$(X):E?E%=X:NEXTE%
740 INPUT#I,X:IFX>ØTHENU%(X)=ZZ+1:
    GOTO74Ø
750 CLOSE#I:RETURN
760 IFAA=ØTHEN8ØØELSEJM=FNP(Ø):CLS:
    GOSUB85Ø
770 FORA%=1TOAA:X=A?A%:GOSUB 83Ø
780 Y=Y+1-(LEN(U$(X))>12):IFY>2Ø
    AND(A%<AA)THENVDU3:GOSUB84Ø:
    GOSUB85Ø
790 NEXTA%:VDU3:GOSUB84Ø
800 IFEE=ØTHENRETURN□ELSE□VDUJM,
    1Ø,1Ø:GOSUB87Ø:FORE%=1TOEE:
    X=E?E%:PRINT;U%(X)TAB(1Ø)U$(X)
810 Y=Y+1-(LEN(U$(X))>12):IFY>2Ø
    AND(E%<EE)THENVDU3:GOSUB84Ø:
    GOSUB87Ø:VDUJM
820 NEXTE%:VDU3:RETURN
830 PRINT;-FNU(S?X);TAB(6);-FNU(F?X);
    TAB(12);T(X);TAB(18);N(X);TAB(23);U%
    (X);TAB(28)U$(X):RETURN
840 INPUT"PRESS RETURN TO CONTINUE",
    F$:CLS:RETURN
850 VDUJM:PRINT"START FINISH TIME 9Ø%
    □□CODE TEXT"
860 PRINT"EVENT EVENT ALLOW SURE":
    Y=3:RETURN
870 PRINT"CODE□□□□□TEXT":Y=3:
    RETURN
880 CK=TRUE:FORA%=1TOAA:X=A?A%
890 Z=S?X:IFS?Z<ØORZZ<U%(Z)THEN
    PRINT;U%(X);W$(5);U%(Z):CK=FALSE
900 Z=F?X:IF(S?Z<ØORZZ<U%(Z))THEN
    PRINT;U%(X);W$(5);U%(Z):CK=FALSE
910 NEXTA%:IF(CK=FALSE)THEN135Ø
920 E%=1
930 Z=E?E%:IFS?Z<ØTHENGOSUB56Ø:IF
    (E%<EE)THEN93Ø
940 E%=E%+1:IF(E%<=EE)THEN93Ø
950 FORE%=1TOEE:Z=E?E%:S?Z=Ø:
    F?Z=Ø:NEXTE%
960 FORA%=1TOAA:X=A?A%:S?(F?X)=X:
    NEXTA%
970 SE=Ø:FORE%=1TOEE:Z=E?E%:
    IFS?Z>ØTHEN1ØØØ
980 IFSE=ØTHENSE=Z:GOTO1ØØØ
990 PRINTW$(1);U%(Z):IF(SE<=MH)THEN
    PRINTW$(1);U%(SE):SE=MH+1
1000 NEXTE%:IFSE=ØTHENPRINT"ALL
    EVENTS HAVE PRECEDING";A$
1010 IFSE=ØOR(SE>MH)THEN135Ø
1020 FORE%=1TOEE:Z=E?E%:T(Z)=Ø:N
    (Z)=Ø:NEXTE%:T(SE)=1
1030 LAST=1:FORC=2TOEE+2:IF
    LAST<>C-1THEN1Ø9Ø
1040 FORA%=1TOAA:X=A?A%:Y=S?X:IFT
    (Y)<>C-1THEN1Ø8Ø
1050 IF(Y=F?X)THENGOSUB114Ø:GOTO1Ø8Ø
1060 IF(Y<>SE)THENY=S?Y:GOTO1Ø5Ø
1070 Y=F?X: S?Y=S?X: F?(S?Y)=Y:
    T(Y)=C: FE=Y: LAST=C
1080 NEXTA%
1090 NEXTC:PRINT"START EVENT=";U%
    (SE);",END EVENT";U%(FE):DD=INKEY
    (3ØØ)
1100 FORE%=1TOEE:Y=E?E%
1110 IFF?Y=ØAND(Y<>FE)THENPRINTU%
    (Y);"NOT LINKED TO END EVENT":CK=
    FALSE
1120 NEXTE%:IF(CK)THEN118Ø
1130 GOTO135Ø
1140 CLS:PRINT"THERE IS A LOOP AS
    FOLLOWS""EVENTS...":XA=A?A%
1150 X=F?XA:PRINTU%(X):Y=S?XA:
    PRINTU%(Y)
1160 Y=S?Y:PRINTU%(Y):IFY<>X THEN
    121Ø
1170 RETURN
1180 K=1:AK=AA:IFAA=1THENK=Ø
1190 AK=INT((AK+K)/2):IFAK=ØTHEN124Ø
1200 K=Ø:FORA%=AK+1TOAA:
    B=A%-AK:X=A?A%:Y=A?B:XE=S?X:
    YE=S?Y
```

```
1210 IFT(YE)+YE/ZZ<=XE/ZZ+T(XE)
   THEN1230
1220 A?A%=Y:A?B=X:K=1
1230 NEXTA%:GOTO1190
1240 N(FE)=LAST:FORD=LAST-1TO
   1STEP-1
1250 FORA%=1TOAA:X=A?A%:IFN(F?X)
   <>D+1THEN1270
1260 Y=S?X:F?Y=F?X:N(Y)=D
1270 NEXTA%:NEXTD
1280 FORA%=1TOAA:G?A%=A?A%:NEXT
   A%:K=1:AK=AA:IFAA=1THENK=Ø
1290 AK=INT((AK+K)/2):IFAK=ØTHEN1340
1300 K=Ø:FORA%=AK+1TOAA:B=A%-
   AK:X=G?A%:Y=G?B:XE=F?X:YE=F?Y
1310 IFN(YE)+YE/ZZ<=XE/ZZ+N(XE)
   THEN1330
1320 G?B=X:G?A%=Y:K=1
1330 NEXTA%:GOTO1290
1340CK=TRUE:RETURN
1350 CK=FALSE:FORX=1TO1000:NEXTX:
   RETURN
1360 IF AA=Ø OR EE=Ø THEN□RETURN
   □ELSE IF NOT(CK) THEN GOSUB880
1365 IF NOT CK THEN RETURN
1370 FORA%=1TOAA:X=A?A%:Z(X)=T(X):
   NEXTA%:GOSUB1440
1380 FORA%=1TOAA:X=A?A%:Y(X)=-(Z
   (F?X)-Y(S?X)=Z(X))*100:NEXTA%
1390 FORB=1TO(AA)STEP5:CLS:
   FORA%=(B)TO FNA(B+4,AA):X=A?A%
1400 PRINTA$;"□";U%(X);"=";U$(X)
1410 C=Y(S?X):D=Z(F?X):PRINT"ABLE TO
   START□";INT(C*100)/100;"□MUST
   FINISH□";INT(D*100)/100
1420 PRINT"SLACK□";INT((D-C-Z(X))*
   100)/100;"CRITICAL□";Y(X);"%":
   IFT=9THENPRINT"STD DEVN=";INT
   (Q(X)*100)/100
1430 PRINT:NEXTA%:GOSUB840:NEXTB:
   RETURN
1440 FORE%=1TOEE:Y(E?E%)=Ø:NEXTE%
1450 FORA%=1TOAA:X=A?A%:Y(F?X)=
   FNZ(Y(F?X) ,Y(S?X)+Z(X)):NEXTA%
1460 FORE%=1TOEE:Z(E?E%)=Y(FE):
   NEXTE%:FORA%=(AA)TO1STEP-1:
   X=G?A%
1470 Z(S?X)=FNA(Z(S?X), Z(F?X)-Z(X)):
   NEXTA%:RETURN
1480 IF NOT(CK)THEN GOSUB880
1485 IF NOT CK THEN RETURN
1490 FORA%=1TOAA:X=A?A%:P(X)=Ø:Q
   (X)=Ø:Y(X)=Ø:NEXTA%
1500 FORE%=1TOEE:Z=E?E%:P(Z)=Ø:Q
   (Z)=Ø:NEXTE%
1510 FORM=1TO43STEP3:FORA%=1TO
   AA:W(A%)=2*RND(1)-1:NEXTA%
1520 FORN=ØTO4STEP2:CLS:PRINT
   "STARTING CASE□";M+N/2;"□OF 45"
1530 FORA%=1TOAA:X=A?A%:TX=T(X):
   IFTX=ØTHENZ(X)=Ø:GOTO1580
```

```
1540 NX=N(X):IF(NX=TX)THENZ(X)=TX:
   GOTO1580
1550 W=FNW(W(A%)+N/3):IFNX>=TX*3
   THENZ(X)=-NX*(W<TX/NX):GOTO 1580
1560 IFNX>TX*2.34THENZ(X)=-TX*LOG
   (W):GOTO1580
1570 W=FNX(W-.5):Z(X)=ABS(TX+W*
   (NX-TX))
1580 NEXTA%
1590 GOSUB1440
1600 FORA%=1TOAA:X=A?A%:Z=Z
   (F?X)-Y(S?X)-Z(X)
1610 P(X)=P(X)+Z:Q(X)=Q(X)+Z*Z:Y(X)
   =Y(X)+(Z<1E-6):NEXTA%
1620 FORE%=1TOEE:Z=E?E%:P(Z)=P(Z)
   +Y(Z):Q(Z)=Q(Z)+Z(Z):NEXTE%:
   NEXTN:NEXTM
1630 FORE%=1TOEE:Z=E?E%:Y(Z)=VAL
   (LEFT$(STR$(P(Z)/45) ,6))
1640 Z(Z)=VAL(LEFT$(STR$(Q(Z)/45) ,6)):
   NEXTE%
1650 FORA%=1TOAA:X=A?A%:Y=Y(X):Y
   (X)=-VAL(LEFT$(STR$(Y/45*100) ,4))
1660 IFP(X)<1.E-2THENP(X)=Ø
1670 Z=(45-Y)+.1E-9:Z(X)=Z(F?X)-Y
   (S?X)-VAL(LEFT$(STR$(P(X)/Z),6))
1680 Q(X)=SQR(ABS((Q(X)-P(X)*P(X)/Z)/
   ((Z-1)+.1E-9))):IFQ(X)<1.E-6THEN
   Q(X)=Ø
1690 NEXTA%:GOTO1390
1700 DEFFNA(X,Y)=(X+Y-ABS(X-Y))/2
1710 DEFFNZ(X,Y)=(X+Y+ABS(X-Y))/2
1720 DEFFNW(X)=-ABS(X)*(X<1)-ABS
   (2-X)*(X>1)
1730 DEFFNX(X)=X*(2.37572+X*X*
   (15.9402-X*X*(184.744-X*X*688.472
   )))/1.20667
1740 DEFFNU(X)=-U%(ABS(X*
   (X<=MH))-(X=ØORX>MH))*(X>Ø
   ANDX<MH)
1750 DEFFNP(JM):PRINT'"DO YOU WANT A
   PRINTOUT(Y/N)?";:REPEATJM=GET:
   UNTILINSTR("YyNn",CHR$(JM)):=3+
   (JM=89ORJM=121)
```

```
590 IFF$<>A$THEN680
600 FORB=1TOAA:IFX=A(B)THENA=B
610 NEXTB:A(A)=A(AA):U(X)=ZZ+1:
   AA=AA-1:RETURN
620 IFU(X)<ØGOSUB1050:PRINTUSING
   "#####□□";ABS(U(X));:PRINT
   U$(X):GOTO650
630 IFEE=ME THENPRINTW$(2);F$:RETURN
640 GOSUB660
650 PRINTW$(4);F$:INPUTU$(X):S(X)=Ø:
   RETURN
660 EE=EE+1:E(EE)=X:S(X)=-1:
   F(X)=Ø:U(X)=U
670 T(X)=Ø:N(X)=Ø:U$(X)="":RETURN
680 Z=X:FORF=1TOEE:IFE(F)=Z THEN E=F
690 NEXTF:E(E)=E(EE):U(Z)=ZZ+1:EE=
   EE-1:RETURN
700 Z=U-INT((U-1)/MH)*MH:Y=2:X=Ø
710 IFX=ØAND(Ø=U(Z)ORZZ+1=U(Z))
   THENX=Z
720 IFU=U(Z)THENX=Z:RETURN
730 IFY=1ORØ=U(Z)THENRETURN
740 Z=Z+Y-MH*INT((Z+Y-1)/MH):
   Y=Y+Y-MH*INT((Y+Y-1)/MH):
   GOTO710
750 OPEN"O",#-1,F$:PRINT#-1,MA;
   ME;MH;AA;EE;CK
760 IFCK THENPRINT#-1,SE;FE
770 FORA=1TOAA:X=A(A):PRINT#-1,X;
   U(X);S(X);F(X);T(X);N(X);G(A);U$(X):NEXTA
780 FORE=1TOEE:Z=E(E):PRINT#-1,Z;
   U(Z);S(Z);F(Z);T(Z);N(Z);U$(Z):NEXTE
790 FORX=1TOMH:IFU(X)=ZZ+1THEN
   PRINT#-1,X
800 NEXTX:PRINT#-1,Ø
810 CLOSE#-1:MOTORON:FORX=1TO
   100:NEXTX:MOTOROFF:RETURN
820 CLS:PRINT"ERROR TRYING TO SAVE
   DATA":FORK=1TO1000:NEXT:RETURN
830 OPEN"I",#-1,F$:INPUT#-1,MA,
   ME,MH,AA,EE,CK:GOSUB20
840 IFCK THENINPUT#-1,SE,FE
850 FORA=1TOAA:INPUT#-1,X,U(X),
   S(X),F(X),T(X),N(X),G(A),U$(X):A(A)=X:
   NEXTA
860 FORE=1TOEE:INPUT#-1,Z,U(Z),S(Z),
   F(Z),T(Z),N(Z),U$(Z):E(E)=Z:NEXTE
870 INPUT#-1,X:IFX>ØTHEN
   U(X)=ZZ+1:GOTO870
880 CLOSE#-1:RETURN
890 GOSUB1870:A=Ø:GOSUB1030
900 FORA=1TOAA:X=A(A):GOSUB1000
910 Y=Y+1:IFY>8ANDA<AA GOSUB
   1020:GOSUB1030
920 NEXTA:GOSUB1020
930 E=Ø:GOSUB1050:FORE=1TOEE:
   X=E(E):PRINT#PR,USING"####
   #□□□";ABS(U(X));:PRINT#PR,U$(X)
940 Y=Y+1:IFY>15ANDE<EE GOSUB
   1020:GOSUB1050
950 NEXTE:GOTO1020
960 CLS:PRINT" ARE YOU SURE (Y/N) ?"
970 T$=INKEY$:IFT$<>"Y"ANDT$<>
   "N"THEN970
980 IFT$="N"THENRETURN
990 CLS:END
1000 PRINT#PR,USING"#####□#
   ####□#####□####
   #□□#####";FNU(S(X)),FNU
   (F(X)),T(X),N(X),ABS(U(X));:IFPR=Ø
   THENPRINT
1010 PRINT#PR,"□";U$(X):RETURN
1020 IFPR=ØTHENPRINT"PRESS ENTER TO
   CONTINUE":INPUTF$:CLS:RETURNELSE
   RETURN
1030 CLS:IFPR=ØORA=ØTHENPRINT#PR,
```

```
   "START FINISH TIME□□□90%
   ACTIVITY":Y=3
1040 RETURN
1050 CLS:IFPR=ØORE=ØTHENPRINT#PR,
   "□□EVENT□□TEXT":Y=3
1060 RETURN
1070 CK=TR:FORA=1TOAA:X=A(A)
1080 Z=S(X):IFS(Z)<ØORZZ<U(Z)THEN
   PRINTA$;U(X);W$(5);U(Z):CK=FA
1090 Z=F(X):IFS(Z)<ØORZZ<U(Z)THEN
   PRINTA$;U(X);W$(5);U(Z):CK=FA
1100 NEXTA:IFCK=FA THEN154Ø
1110 E=1
1120 Z=E(E):IFS(Z)<ØGOSUB68Ø:IF
   E<=EE THEN112Ø
1130 E=E+1:IFE<=EE THEN112Ø
1140 FORE=1TOEE:Z=E(E):S(Z)=Ø:
   F(Z)=Ø:NEXTE
1150 FORA=1TOAA:X=A(A):S(F(X))=X:
   NEXTA
1160 SE=Ø:FORE=1TOEE:Z=E(E):IF
   S(Z)>ØTHEN119Ø
1170 IFSE=ØTHENSE=Z:GOTO119Ø
1180 PRINTW$(1);U(Z):IFSE<=MH THEN
   PRINTW$(1);U(SE):SE=MH+1
1190 NEXTE:IFSE=ØTHENPRINT"ALL
   EVENTS HAVE PRECEDING";A$
1200 IFSE=ØORSE>MH THEN154Ø
1210 FORE=1TOEE:Z=E(E):T(Z)=Ø:
   N(Z)=Ø:NEXTE:T(SE)=1
1220 LA=1:FORC=2TOEE+2:IF
   LA<>C-1THEN128Ø
1230 FORA=1TOAA:X=A(A):Y=S(X):IF
   T(Y)<>C-1THEN127Ø
1240 IFY=F(X)GOSUB133Ø:GOTO127Ø
1250 IFY<>SE THENY=S(Y):GOTO124Ø
1260 Y=F(X):S(Y)=S(X):F(S(Y))=Y:
   T(Y)=C:FE=Y:LA=C
1270 NEXTA
1280 NEXTC:PRINT"START EVENT ";U(SE);
   ", END EVENT ";U(FE)
1290 FORE=1TOEE:Y=E(E)
1300 IFF(Y)=ØANDY<>FE THENPRINT U(Y)
   ;"NOT LINKED TO END EVENT": CK=FA
1310 NEXTE:IFCK THEN137Ø
1320 GOTO154Ø
1330 CLS:CK=FA:PRINT"THERE IS A LOOP
   AS FOLLOWS":PRINT"EVENTS ... ":
   XA=A(A)
1340 X=F(XA):PRINTU(X):Y=S(XA):PRINT
   U(Y)
1350 Y=S(Y):PRINTU(Y):IFY<>X THEN
   135Ø
1360 FORX=1TO1ØØØ:NEXT:RETURN
1370 K=1:AK=AA:IFAA=1THENK=Ø
1380 AK=INT((AK+K)/2):IFAK=ØTHEN 143Ø
1390 K=Ø:FORA=AK+1TOAA:B=A-AK:
   X=A(A):Y=A(B):XE=S(X):YE=S(Y)
1400 IFT(YE)+YE/ZZ<=XE/ZZ+T(XE)
   THEN142Ø
1410 A(A)=Y:A(B)=X:K=1
1420 NEXTA:GOTO138Ø
1430 N(FE)=LA:FORD=LA-1TO1STEP-1
1440 FORA=1TOAA:X=A(A):IFN(F(X))<>
   D+1THEN146Ø
1450 Y=S(X):F(Y)=F(X):N(Y)=D
1460 NEXTA:NEXTD
1470 FORA=1TOAA:G(A)=A(A):NEXTA:
   K=1:AK=AA:IFAA=1THENK=Ø
1480 AK=INT((AK+K)/2):IFAK=ØTHEN
   153Ø
1490 K=Ø:FORA=AK+1TOAA:B=A-AK:
   X=G(A):Y=G(B):XE=F(X):YE=F(Y)
1500 IFN(YE)+YE/ZZ<=XE/ZZ+N(XE)
   THEN152Ø
1510 G(B)=X:G(A)=Y:K=1
1520 NEXTA:GOTO148Ø
1530 CK=TR:RETURN
1540 CK=FA:FORX=1TO1ØØØ:NEXTX:
   RETURN
1550 GOSUB187Ø:FORA=1TOAA:X=A(A):
   Z(X)=T(X):NEXTA:GOSUB162Ø
1560 FORA=1TOAA:X=A(A):Y(X)=
   -(Z(F(X))-Y(S(X))=Z(X))*1ØØ:NEXTA
1570 FORB=1TOAA STEP3:CLS:FOR A=B□
   TOAA+FNA(B+2-AA):X=A(A)
1580 PRINT#PR,A$;U(X);"=";U$(X)
1590 C=Y(S(X)):D=Z(F(X)):PRINT#PR,
   "CAN START";C;"MUST END";D
1600 PRINT#PR,"SLACK";INT(1ØØ*
   (D-C-Z(X)))/1ØØ;"(CRITICAL";Y(X);
   "%)":IFT=9THENPRINT#PR,USING
   "STD DEVN=####.##";Q(X)
1610 PRINT#PR:NEXTA:GOSUB1Ø2Ø:NEXTB:
   RETURN
1620 FORE=1TOEE:Y(E(E))=Ø:NEXTE
1630 FORA=1TOAA:X=A(A):Y(F(X))=
   Y(F(X))+FNZ(Y(S(X))-Y(F(X))+Z(X)):
   NEXTA
1640 FORE=1TOEE:Z(E(E))=Y(FE):NEXTE:
   FORA=AA TO1STEP-1:X=G(A)
1650 Z(S(X))=Z(S(X))+FNA(Z(F(X))-
   Z(S(X))-Z(X)):NEXTA:RETURN
1660 GOSUB187Ø:FORA=1TOAA:X=A(A):
   P(X)=Ø:Q(X)=Ø:Y(X)=Ø:NEXTA
1670 FORE=1TOEE:Z=E(E):P(Z)=Ø:
   Q(Z)=Ø:NEXTE
1680 FORM=1TO43STEP3:FORA=1TOAA:
   W(A)=2*RND(Ø)-1:NEXTA
1690 FORN=ØTO4STEP2:CLS:PRINT
   "STARTING CASE";M+N/2;" OF 45"
1700 FORA=1TOAA:X=A(A):TX=T(X):IF
   TX=ØTHENZ(X)=Ø:GOTO175Ø
1710 NX=N(X):IFNX=TX THENZ(X)=TX:
   GOTO175Ø
1720 W=FNW(W(A)+N/3):IFNX>=TX*3
   THENZ(X)=-NX*(W<TX/NX):GOTO
   175Ø
1730 IFNX>TX*2.34THENZ(X)=-TX*LOG
   (W):GOTO175Ø
1740 W=FNX(W-.5):Z(X)=ABS(TX+W*
   (NX-TX))
1750 NEXTA
1760 GOSUB162Ø
1770 FORA=1TOAA:X=A(A):Z=Z(F(X))-
   Y(S(X))-Z(X)
1780 P(X)=P(X)+Z:Q(X)=Q(X)+Z*Z:
   Y(X)=Y(X)+(Z<1E-6):NEXTA
1790 FORE=1TOEE:Z=E(E):P(Z)=P(Z)+
   Y(Z):Q(Z)=Q(Z)+Z(Z):
   NEXTE,N,M
1800 FORE=1TOEE:Z=E(E):Y(Z)=VAL
   (LEFT$(STR$(P(Z)/45),6))
1810 Z(Z)=VAL(LEFT$(STR$(Q(Z)/45),6)):
   NEXTE
1820 FORA=1TOAA:X=A(A):Y=Y(X):
   Y(X)=-VAL(LEFT$(STR$(Y/45*1ØØ),4))
1830 IFP(X)<1E-2THENP(X)=Ø
1840 Z=45-Y+.1E-9:Z(X)=Z(F(X))-
   Y(S(X))-VAL(LEFT$(STR$(P(X)/Z),6))
1850 Q(X)=SQR(ABS((Q(X)-P(X)*P(X)/Z)/
   ((Z-1)+.1E-9))):IFQ(X)<1.E-6THEN
   Q(X)=Ø
1860 NEXTA:GOTO157Ø
1870 IF(PEEK(65314)AND1)=1THENRETURN
   ELSECLS:PRINT"SEND TO PRINTER OR
   SCREEN (P/S)?"
1880 Q$=INKEY$:IFQ$<>"P"ANDQ$
   <>"S"THEN188Ø
1890 IFQ$="P"THENPR=-2
1900 CLS:RETURN
```

Dragon users with a Dragon Data disk drive should make these changes:

```
750 ERROR GOTO82Ø:CREATE F$:
   FWRITE F$;MA;E$;ME;E$;MH;E$;
   AA;E$;EE;E$;CK
760 IFCK THENFWRITEF$;SE;E$;FE
770 FORA=1TOAA:X=A(A):FWRITEF$;X;E$;
   U(X);E$;S(X);E$;F(X);E$;T(X);E$;N(X);E$;
   G(A);E$;U$(X):NEXTA
780 FORE=1TOEE:Z=E(E):FWRITEF$;Z;E$;
   U(Z);E$;S(Z);E$;F(Z);E$;T(Z);E$;N(Z);E$;
   U$(Z):NEXTE
790 FORX=1TOMH:IFU(X)=ZZ+1THEN
   FWRITEF$;X
800 NEXTX:FWRITEF$;Ø
810 CLOSE:RETURN
820 CLS:PRINT"ERROR TRYING TO SAVE
   DATA":FORK=1TO1ØØØ:NEXT:
   RETURN
830 FREADF$,FROMØ;MA,ME,MH,AA,EE,CK:
   GOSUB2Ø
840 IFCK THENFREADF$;SE,FE
850 FORA=1TOAA:FREADF$;X,U(X),S(X),
   F(X),T(X),N(X),G(A),U$(X):A(A)=X:
   NEXTA
860 FORE=1TOEE:FREADF$;Z,U(Z),S(Z),
   F(Z),T(Z),N(Z),U$(Z):E(E)=Z:
   NEXTE
870 FREADF$;X:IFX>ØTHENU(X)=ZZ+1:
   GOTO87Ø
880 CLOSE:RETURN
```

A PICTURE TEST CARD PROGRAM

If you get eyestrain when you're fiddling with your hardware, you could be due for a change of monitor. This test program sorts the flicker from the fuzz

Until quite recently the price of a monitor has been well outside the pocket of most home computer owners and, in general, people have had to make do with the family TV set. But the ordinary TV has several limitations, and if you have been using one for some time you are probably only too well aware of them. The picture on a TV set is continuously flickering and juddering, and this can cause eye strain and headaches if you have to look at it for long periods—if you are doing any wordprocessing, for instance. Also, the resolution is usually quite low which means detail is lost in graphics and games. And, of course, the TV is often hijacked by the rest of the family who want to watch the news or a film.

But now that the price of monitors has started to fall, and good quality combined TV/monitors have been introduced, you may be considering buying a set to use specifically with your computer. The question is, how do you choose which to buy?

The article on page 445 should help you make your choice as it describes the differences between a TV and a monitor and explains such technical terms as bandwidth, signal types and so on. But in the end, after you've sorted out all the technical details, the important point is what the picture looks like to you, when connected up to your computer. It is essential to use your own computer in these tests since not all TVs and monitors are compatible with all computers, and even computers of the same make can produce quite different results. So take your computer to the shop and test the complete system.

The test card program given below is designed to help you assess different systems. It is not an absolute test but it does give you a fairly objective way of comparing several TVs or monitors—and it is short enough to type in in the shop.

A SCREEN TEST

The Spectrum, Commodore and Vic programs print a single, combined test card, while the Acorn, Dragon and Tandy programs use several routines to test different modes. Here are the points to look out for.

First check whether the picture is centred in the screen. This is most critical on the Acorns as the computer doesn't leave a border round the picture and some text may be lost. Also on the Acorns check if the characters in the corners are legible—or, indeed, if any characters are legible.

Now have a look at the rectangles or the edge of the test card. On most sets the lines near the edge of the screen bulge outwards.

Are all of the lines clearly visible, or do they merge together? This is a good test of the resolution. On most colour TVs and monitors the resolution is better in one direction than the other due to the way the coloured phosphor dots are arranged on the screen, but some monitors use a different arrangement of dots and give a clearer picture.

If the program draws circles are they really circular or are they oval?

Now check the colour. The edges of the coloured squares should be sharp, not fuzzy, and the colours shouldn't run into each other. The coloured text on the different coloured background should be clearly readable. (On some sets, with some colour combinations, it may be completely illegible.) The density of the colour may vary, too, from set to set so try to decide which you prefer.

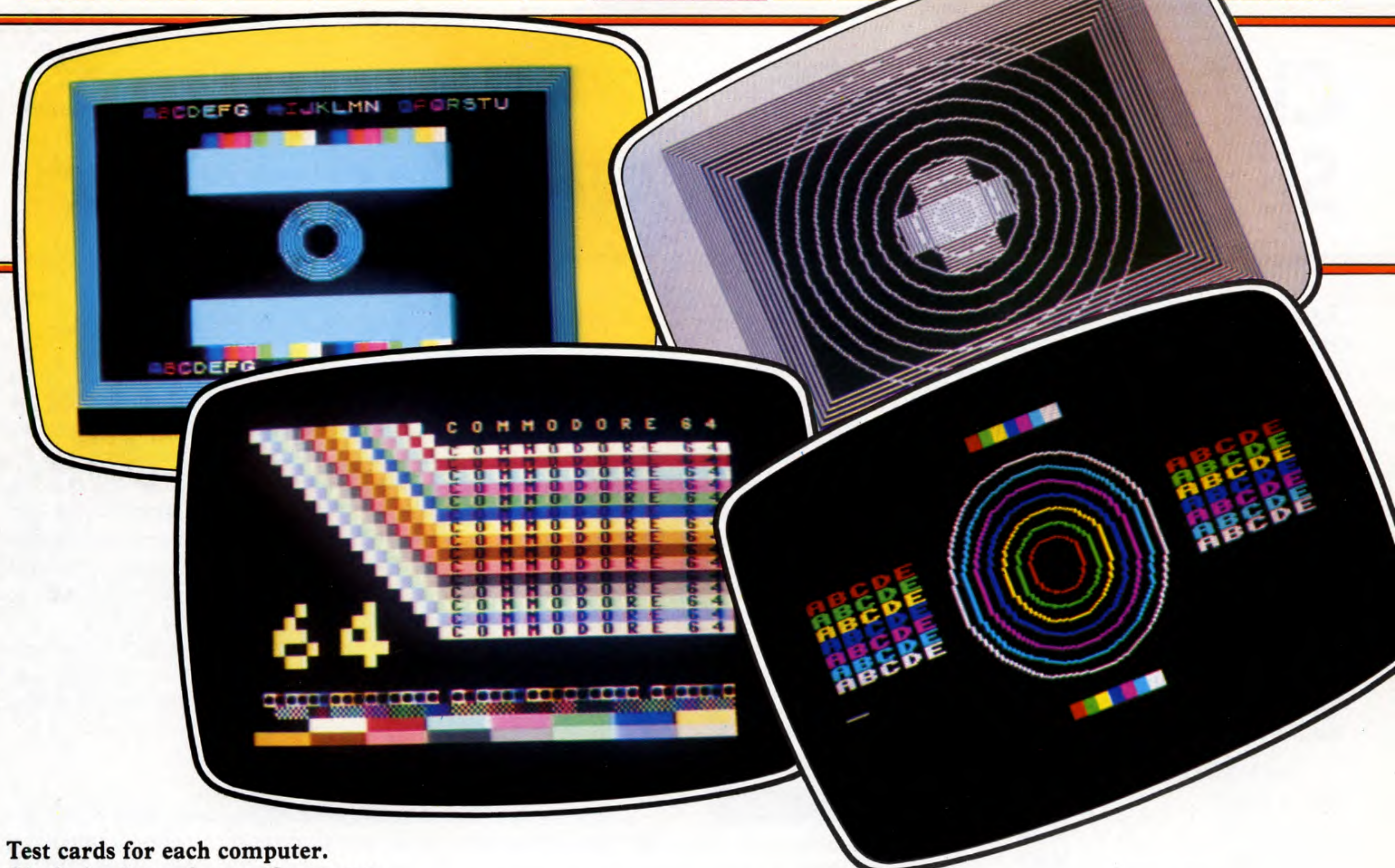
Lastly, check the whiteness of the lines or text to see if there's any colour fringing.

Of course, the final deciding point may be the cost, but at least you'll be in a position to choose the best in your price range.

```
5 FOR p=0 TO 7: FOR i=0 TO 7: IF p=i
   THEN GOTO 200
6 PAPER p: INK i: BORDER 0: CLS
10 FOR n=0 TO 12 STEP 2: PLOT n,n: DRAW
   0,175-2*n: DRAW 255-2*n,0: DRAW
   0,-(175-2*n): DRAW -(255-2*n),0:
   NEXT n
20 FOR n=10 TO 20 STEP 2: CIRCLE INK
   i;128,85,n: NEXT n
30 FOR n=5 TO 7: PRINT INK i;AT n,8;"";AT
   n+10,8;"": NEXT n
40 PRINT AT 4,8;: FOR m=0 TO 1: FOR
   n=0 TO 7: PRINT PAPER n;"□";: NEXT
   n: NEXT m
50 PRINT AT 18,8;: FOR m=0 TO 1: FOR
   n=0 TO 7: PRINT PAPER n;"□";: NEXT
   n: NEXT m
55 PRINT AT 2,4;: FOR n=0 TO 2: FOR
   m=0 TO 7: PRINT INK m; PAPER p;CHR$
   (64+n*7+m);: NEXT m: NEXT n
57 PRINT AT 19,4;: FOR n=0 TO 2: FOR
   m=0 TO 7: PRINT BRIGHT 1; INK m;
   PAPER p; CHR$ (64+n*7+m);: NEXT m:
   NEXT n
60 FOR n=0 TO 7: BORDER n: PAUSE 50:
   NEXT n: PAUSE 50
200 NEXT i: NEXT p
```

```
10 PRINT "■":Z$="■■■":ZZ$=
   "□C□O□M□M□O□D□O□R□E
   □□6□4□"
20 X=1040:X1=1944:X2=1904:FOR Z=0
   TO 15:X=X-1:XX=X
30 FOR ZZ=0 TO Z:POKE XX,160:POKE
   54272+XX,Z:XX=XX+41:NEXT ZZ
35 POKE 646,Z:PRINT SPC(16)"■"ZZ$;:Z$
   =Z$+"■"
40 FOR Z1=0 TO 4:POKE X1,160:POKE
   54272+X1,Z:POKE X2,102:POKE
   54272+X2,Z
50 X1=X1+1:X2=X2+.5:NEXT Z1,Z
60 A$="■■■■■■□■■■■□■
   ■■■□□□■■■■■■□■■□
   ■■■■■□□"
70 B$="■■■■■■□■■■■□□
   ■■■■■□■□■■■■■■□□
   □□■■■■□"
75 PRINT Z$"■■■■":FOR Z=0 TO
   39:POKE 646,Z:PRINT "■●";:NEXT Z
80 FOR Z=0 TO 15:POKE 646,Z:PRINT Z$A$
   Z$SPC(6)B$
85 PRINT "■"SPC(16)ZZ$
90 GET K$:IF K$="■" THEN C1=C1+1
100 IF K$="■" THEN C2=C2+1
110 C1=C1 AND 15:C2=C2 AND 15:POKE
   53280,C1:POKE 53281,C2
120 NEXT Z:GOTO 80
```

```
10 PRINT "■":Z$="■■":ZZ$="□
   V□I□C□□2□0□":C1=44
20 X=7690:X1=8010:X2=7988:FOR Z=0
   TO 7:X=X-1:XX=X
30 FOR ZZ=0 TO Z:POKE XX,160:POKE
   30720+XX,Z:XX=XX+23:NEXT ZZ
35 POKE 646,Z:PRINT SPC(10)"■"ZZ$;:Z$
   =Z$+"■"
```



Test cards for each computer. Clockwise from the top, Spectrum, Acorn, Dragon and Commodore

```
40 FOR Z1=Ø TO 21:POKE X1,16Ø:POKE
   3Ø72Ø+X1,Z:POKE X2,1Ø2:POKE 3Ø72Ø
   +X2,Z
50 X1=X1+1:X2=X2+.12:NEXT Z1,Z
60 A$="○○■■□■■■■□■□
   ■■□■■■■■■■■□■■□
   □□"
70 B$="○○■■□■■■■□■□
   ■■■■■□■□■■■■■□■
   □■■■■■□■□■■■■□"
75 PRINT Z$"■■■■":FOR Z=Ø TO
   21:POKE 646,Z AND 7:PRINT "■●";:
   NEXT Z
80 FOR Z=Ø TO 7:POKE 646,Z:PRINT Z$A$
   Z$SPC(4)B$
85 PRINT "■"SPC(1Ø)ZZ$
90 GET K$:IF K$="■" THEN C1=C1+1
110 C1=C1 AND 255:POKE 36879,C1
120 NEXT Z:GOTO 8Ø
```

```
10 MODEØ
20 FOR T=1 TO 1ØØØ:VDU RND(96)+31:
   NEXT
30 D=GET:CLS: S=2
40 FOR T=Ø TO 52Ø STEP S
50 MOVET,T:DRAW1276-T,T:DRAW1276-T,
   1Ø23-T:DRAWT,1Ø23-T:DRAWT,T
60 T=T+(T□DIV 5Ø):NEXT
100 D=GET
110 FOR Q=Ø TO 7:MODE2:GCOLØ,128+Q:
   CLG:COLOUR128+Q
120 Y=824
130 FOR T=1 TO 2
140 X=64Ø-4*32
150 FOR C=Ø TO 7
160 GCOLØ,C
170 PROCB
180 X=X+32
190 NEXT
200 Y=168
210 NEXT
220 VDU 29,64Ø;512;
230 W=Ø
240 FOR R=32 TO 256 STEP 32
250 GCOLØ,W:W=W+1
260 PROCC
270 NEXT
280 FOR T=Ø TO 7
290 COLOURT
300 PRINTTAB(Ø,T+11)"ABCDE"TAB(15,
   T+11)"ABCDE"
310 NEXT
320 D=INKEY(3ØØ):NEXT
330 END
340 DEF PROCB
350 MOVEX,Y:MOVEX+32,Y:PLOT85,X,
   Y+32:PLOT85,X+32,Y+32
360 ENDPROC
370 DEF PROCC
380 LOCALT
390 MOVE COS(PI*2+.15)*R,Ø
400 FORT=-.1 TO PI*2+.15 STEP .15
410 DRAW COS(T)*R,SIN(T)*R
420 NEXT: ENDPROC
```

```
10 PMODE4,1:PCLS:SCREEN1,1
20 FORK=ØTO1:FORJ=ØTO15 STEPK+2
30 LINE(K*16+J,K*16+J)-(255-K*
   16-J,191-K*16-J),PSET,B
40 NEXTJ,K
50 FORK=1ØØTO154 STEP2:LINE(K,85)
   -(K,1Ø5),PSET:LINE(113,K-32)-(141,
   K-32),PSET:NEXT
60 FORK=1TO1ØØ STEP 1Ø:CIRCLE(127,95),
   K,5:NEXT
70 IFINKEY$="" THEN7Ø
80 PMODE1,1:PCLS:SCREEN1,Ø
90 FORJ=ØTO191 STEP48:COLORJ/48+1
100 LINE(Ø,J)-(255,J+47),PSET,BF:NEXT
110 FORJ=ØTO255 STEP63:FORK=ØTO63
   STEP4:COLORJ/63+1
120 LINE(K+J,Ø)-(K+J,191),PSET
130 NEXTK,J
140 IFINKEY$="" THEN14Ø
150 S=1-S:SCREEN1,S:GOTO14Ø
```

CLIFFHANGER: SETTLING THE SCORE

Not everything that happens to Willie is bad. Sometimes—with your able assistance—he wins through to his reward, retrieves his picnic and clocks up more score

All sorts of things have happened to Willie. He has fallen down holes, been bitten by snakes, hit by boulders and drowned by the sea. And in the last part of Cliffhanger Willie was killed and buried and sent down to Hades.

But now it is time for Willie to get his reward. Here he finally reaches his goal, manages to retrieve an item of his picnic and increment his score.

This first little routine sounds the reward bell, puts Willie up onto the next level, speeds the game up and gives him a massive boost to his score.

```
    org 59788
rwd ld de,523
    ld hl,8Ø6
    call 949
    ld a,(57344)
    inc a
    res 2,a
    ld (57344),a
    ld a,(58732)
    dec a
    ld (58732),a
    ld a,2
    ld b,5
    call 599ØØ
    jp 586Ø1
```

The first three instructions sound the reward bell. This is done using the **BEEPER** routine, at 949 in the Spectrum's ROM, and the pitch and duration parameters are fed to it in the usual way, via the DE and the HL register pairs.

ON THE LEVEL

The level number is then loaded from its storage location at 57,334 into the accumulator. The contents of the accumulator are then incremented.

But there are only four levels in the game, so you don't want the level number to get any bigger than 3. This is prevented by using the **res 2,a** instruction which **res**ets bit **2** of the **a**ccumulator. When level number 3 is incremented to 4, bit 2 is set. Resetting it returns the level number to Ø and puts the game back on level one.

The result of these operations is stored by 57,334 when it can be referred to when setting up the game.

ABOUT SPEED

The delay at 58,732 is then reduced by one. Its value is loaded into the accumulator, decremented and stored back in 58,732. This speeds the game up as the processor does not pause so long in the main routine.

It is originally set to 5Ø, so as long as Willie does not reach the reward more than fifty times the game will go on getting faster and faster. This makes the game more and more difficult. Even though you will be performing the same four levels over and over again, each time they will get faster.

PICKING UP POINTS

For reaching the reward, Willie picks up an extra 500 points. This is done by calling the score routine at 59,9ØØ and feeding parameters to it in the A and B registers.

The 2 loaded into A specifies that it is the digit in the second column from the left—the hundreds—is to be incremented. And the 5 in B tells the routine how many times to increment that digit. Incrementing the hundreds five times increases the score by 500.

The processor then jumps back to the 'new life routine'—labelled **nlv**—at 58,6Ø1 and starts Willie off again at the bottom of the slope.

SCORING

The next little routine keeps track of Willie's score and increments it when he reaches a reward or scales another part of the cliff.

```
    org 599ØØ
scn ld ix,57337
    ld d,Ø
    ld e,a
    add ix,de
scr push ix
    call sdi
    pop ix
    djnz scr
    call 58939
    ret
sdi ld a,(ix+Ø)
    inc a
    cp 1Ø
    jr nz,sno
    ld a,Ø
    ld (ix+Ø),a
    dec ix
    jr sdi
sno ld (ix+Ø),a
    ret
```

- MOVING UP A LEVEL
- SPEEDING UP THE GAME
- ADDING SCORE
- FINDING YOUR PLACE
- DEALING WITH CARRIES

The 'CLIFFHANGER' listings published in this magazine and subsequent parts bear absolutely no resemblance to, and are in no way associated with, the computer game called 'CLIFF HANGER' released for the Commodore 64 and published by New Generation Software Limited.

Memory location 57,337 is the start of the score variable which is loaded with zeros in the initialization routine on page 1101. D is set with zero and E is loaded with the contents of the accumulator. Remember, the accumulator carries the column number when the processor enters this routine.

The contents of DE are then added to those of IX and the result is left in IX. This effectively moves the data pointer along the digits, which are stored in 57,337 onwards, until it gets to the one specified by the contents of A. This position is stored temporarily by pushing the contents of IX onto the stack. The **sdi** routine is then called.

DEALING WITH DIGITS

The **sdi** routine is the one that actually deals with the digits. It starts off loading the accumulator with the digit pointed to by IX. The Ø offset is required here because of the format of the instruction.

The digit is then incremented and compared to 1Ø. If it has reached 1Ø, you will have to increment the next digit too. So on a non-zero result—in other words, the first digit hasn't been incremented to 1Ø yet—the **jr nz** instruction sends the processor forward to the **sno** label where the incremented digit is stored back in the location it was taken from.

If the digit you're dealing with has been incremented up to 1Ø, the jump does not occur and the processor continues with the next instruction. Zero is then loaded into Ø and stored back in the appropriate digit. Then IX is decremented so that it points to the next digit to the left. The **jr sdi** then sends the processor back to **sdi** to start the incrementing routine all over again, on the next digit.

If the next digit then increments to 1Ø you go round the loop again. And so on. But sooner or later, one digit will not overflow,

the processor will get to **sno**, store the last digit and return to the place in the **scr** routine where **sdi** was called.

MORE SCORE

After the processor returns, the IX pointer is popped off the stack again. You can now see why it was stored there. If the digit had been incremented to 1Ø, the **sdi** would have shifted this pointer onto the next digit. And if you wanted the increment again, you'd be doing it to the wrong digit.

The loop here is closed by a **djnz**, which decrements the contents of the B register and jumps if it hasn't been decremented down to zero. B, you'll remember, carried the number of times the score digit had to be incremented when the processor entered the routine.

So the processor goes round this loop, clocking up the score, the number of times B carries to start with. And when it has counted down to zero, the score-printing routine at 58,939 is called which prints up the new score on the screen.

That done, the processor returns.

The following little routine increments the score by one point:

```
      ORG 26624
      LDX #5
NEXT  LDA $Ø47E,X
      CMP #57
      BEQ ZERO
      INC $Ø47E,X
      RET RTS
ZERO  LDA #48
      STA $Ø47E,X
      DEX
      JMP NEXT
```

The score on the screen is five digits long, so the index register, X, is loaded with 5. The contents of X are then used as an offset in the indexed instruction LDA $Ø47E,X. This loads up the digit pointed to by 115Ø plus X on the screen. On entering this routine X is 5 so the units' digit is loaded up from $Ø47E.

This is compared to 57, the ASCII for the figure nine. If it is a nine, adding 1 to the score will cause an overflow so the BEQ instruction branches the processor forward. If not, the contents of $Ø47E plus X are incremented and the processor returns.

DOUBLE DIGITS

If there is going to be an overflow and the next digit to the left needs to be incremented too, the processor loads A with 48. This is the ASCII for a figure zero. The 48 is stored back in $Ø47E plus X on the screen, giving a Ø in that position.

Then X is decremented which moves it effectively one place to the left. And the processor jumps back to the label NEXT.

There it begins to handle the next digit in exactly the same way. If it isn't a figure nine, the digit is incremented. If it is, a figure zero is stored in that location on the screen and the processor goes back to increment the next digit.

The following routine works out the score to base ten, so that it can be printed on the screen, and updates the score each time Willie earns an extra point.

Remember to set up the computer as usual before you start keying it in.


```
30 FORPASS=
   ØTO3STEP3
40 P%=&212E
50 [OPTPASS
60 .Score
70 LDX#6
80 .Lb1
90 LDA&89,X
100 CMP#10
110 BCCLb2
120 SEC
130 SBC#10
140 STA&89,X
150 INC&88,X
160 .Lb2
170 DEX
180 BNELb1
190 RTS
200 .Incsc
210 LDA&7E
220 CMP&7B
230 BCCLb3
240 RTS
250 .Lb3
260 LDA&8F
270 CLC
280 ADC#7
290 STA&8F
300 JSRScore
310 LDA#17
320 JSR&FFEE
330 LDA#128
340 JSR&FFEE
350 JSR&1A3C
360 LDA&7B
370 STA&7E
380 CMP#56
390 BEQLb4
400 RTS
410 .Lb4
420 LDA&8C
430 CLC
440 ADC&83
450 STA&8C
460 INC&8C
470 INC&8C
480 JSR Score
490 INC&83
500 LDA&83
510 CMP#5
520 BNELb5
530 LDA#Ø
540 STA&83
550 INC&89
560 .Lb5
570 LDA&7D
580 ORA#&8Ø
590 STA&7D
600 LDA#15
610 LDX#Ø
620 JSR&FFF4
630 JSR&14E7
640 LDA&84
650 CMP#Ø
660 BEQLb6
670 CLC
680 SBC#2
690 STA&84
700 .Lb6
710 RTS
720 ]NEXT
730 ?&1D65=?&1D62
740 ?&1D90=?&1D8D
```

To test this routine set PAGE=&2500 type NEW and then, key in the following program and RUN it when you have all the other routines in memory.

```
30 CALL&1D77
40 CALL&1D9B
50 CALL&1E99
60 IF?(&1B2D
   +?&83)AND4
   CALL&1DEE
70 REPEAT
80 CALL&1CØ8
90 CALL&1CCB:
   CALL&1CCB:
   CALL&1CCB
95 CALL&11ØØ
100 IF?(&1B2D
   +?&83)AND4
   CALL&1E1D
110 CALL&1FD5
115 IF(?&7C□AND4)
   =ØCALL&1EB6
120 CALL&2ØC9
130 CALL&21Ø3
140 UNTIL?&7
   D□AND128
150 IF?&89=Ø
   REPEATUNTIL
   INKEY(-99):
   RUN ELSEGOTO4Ø
```

LOADING BASE TEN

The score that is printed on the screen is in base ten and has six digits so that it can record any number from zero to 999,999. Although the numbers between zero to 999,999 can be stored in only three bytes ordinarily, they are going to be stored in six bytes here—with one decimal digit in each byte—so that they can be printed up on the screen more easily.

So first X is loaded with 6 so that you can count across the six memory locations that are going to be used. These are zero page locations &8A to &8F. Then A is loaded with the contents of memory location &89 offset by X. In other words, you're going to start at &8F which is the least significant digit. The contents of this memory location are compared with 1Ø.

If the contents of this location have not been incremented up to 1Ø yet, the BCC instruction branches the processor forward over the next routine. But if the contents are 1Ø or larger, the processor continues. Obviously, in base ten notation, you cannot have a number larger than 9 occupying a digit.

CLOCKING ON

If the number in one digit location is 1Ø or over, you have to clock up one in the next digit to the left. But first you need to adjust the contents of the original digit byte.

So the carry flag is set in Line 12Ø and 1Ø is subtracted in Line 13Ø. The result is then stored back in the memory location given by &89 offset by X—which was the one it was loaded up from. The next digit to the left—given by &88 offset by X—is incremented.

The DEX in Line 17Ø decrements X to move onto the next digit. Then the BNE instruction in Line 18Ø branches the processor back to handle the next digit, if X hasn't counted down to zero.

If it has clocked down to zero, all the digits have been dealt with, the processor proceeds, hits the RTS and returns.

WILLIE THE WINNER

Location &7E contains the Y coordinate Willie has just moved from. So this is loaded into the accumulator and compared with the contents of &7B, the Y coordinate Willie has just moved to.

If Willie has not moved up the slope, the BCC instructor does not operate and the processor hits the RTS and returns. But if Willie has moved up the screen, the BCC branches the processor into the next part of the routine.

There, the contents of &8F—which carries the 1s—is loaded up into the accumulator. The carry flag is cleared and 7 is added. The result is stored back in in &8F and the processor jumps to the Score routine—given in the first section of this part of Cliffhanger—which straightens out the decimal digits.

Loading A with 17 and jumping to &FFEE, then loading A with 128 and jumping to &FFEE again, sees the colour for printing up the score. And jumping to the subroutine at &1A3C prints it up on the screen.

Next the Y coordinate counter in &7B is loaded up and stored in &7E—you'll notice that the program has compared the contents of these two locations to see whether Willie has moved. The instruction in Line 38Ø compares this Y coordinate with 56, to see whether Willie has reached the top of the slope where he gets his reward.

If he hasn't the BEQ instruction does not operate and the processor hits the RTS and returns. But if he has hit those heights, the

BEQ instruction takes the processor on into the next routine.

BIG BUCKS

If Willie has reached his reward, this is where the big bucks are scored. When the processor branches on to the label Lb4 in Line 41Ø, it next loads up into the accumulator the contents of &8C, the location that holds the thousands.

The carry flag is then cleared and the contents of &83—the current screen number—are added. The result is stored back in &8C, then &8C is incremented twice for reaching the reward. The Score routine is then called again to straighten up the digits.

Next the contents of &83 are incremented. If Willie had reached the reward, he moves onto the next screen. So the screen number in &83 has to be incremented. Then it is loaded into A by the instruction in Line 5ØØ and compared with 5.

If the screen number has not been incremented to 5—the last screen—yet, the BNE instruction branches the processor forward over the next routine.

Otherwise, A is loaded with Ø which is stored in &83, the level location. This starts the game off again on the next level.

Next, the contents of &89, Willie's lives, are incremented. This adds a life for reaching the reward.

REWARDED

The contents of Willie's status register, &7Ø, are then loaded into the accumulator and ORed with &8Ø. This sets bit seven and tells the processor to bring on a new screen next time this byte is checked—the result of the ORing is stored back in &7Ø.

Next A is loaded with 15 and X with Ø and the subroutine at &FFF4 is jumped to. This the equivalent of a BASIC *FX15,Ø and clears the sound. Then the subroutine at &14E7 is jumped to, to sound the bell.

ON SPEED

Every time Willie reaches a reward and goes up a level, the game is speeded up. So the contents of the location that control the speed, &84, are loaded up into the accumulator and compared with Ø. This checks whether the speed has reached its fastest already.

If it has, the BEQ branches the processor on to the end of the program. If not, the carry flag is cleared and 2 is subtracted to increase the speed.

The result is stored back in &84 and the processor moves on to the RTS, and returns.

The POKEs in Lines 73Ø and 74Ø adjust data given in earlier parts of Cliffhanger.

This little routine sounds the reward bell, puts Willie up onto the next level, speeds the game up and gives him a massive boost to his score, among other things.

```
      ORG 2Ø721
RWD   LDA #255
      LDX #15Ø
      JSR SOUND
      LDA 18238
      INCA
      ANDA #3
      STA 18238
      DEC DLL+1
      LDB #5
      LDA #3
      JSR SCI
      LBRA NLV
SCI   EXG A,B
SCT   LDX #1824Ø
      ABX
      PSHS A,X
      JSR SDI
      PULS A,X
      DECA
      BNE SCT
      JSR PRSC
      RTS
SDI   LDA ,X
      INCA
      CMPA #1Ø
      BNE SNO
      CLR ,X
      LEAX -1,X
      BRA SDI
SNO   STA ,X
      RTS
SOUND PSHS A
      LDA $FFØ1
      ANDA #247
      STA $FFØ1
      LDA $FFØ3
      ANDA #247
      STA $FFØ3
      LDA $FF23
      ORA #8
      STA $FF23
      ORCC #$5Ø
      PULS A
      PSHS X
      LDB #252
SBN   STB $FF2Ø
SC    LEAX -1,X
      BNE SC
      LDX ,S
      CLR $FF2Ø
SD    LEAX -1,X
      BNE SD
      LDX ,S
      DECA
      BNE SBN
      ANDCC #$AF
      PULS X
      RTS
CLICK LDX #98
      LDA #4
      JSR SOUND
      RTS
DLL   EQU $51ED
NLV   EQU $4BF7
PRSC  EQU $4C77
```

The first three instructions sound the reward bell. This is done using the SOUND routine lower down in the program. The pitch and duration parameters are fed to it in the usual way, via the numbers loaded into the A and X registers.

LEVELLING UP

A is loaded with the contents of 18,238, the level's storage location. It is then incremented and ANDed with 3. This clears the six most significant bits and stops the level number being incremented higher than 3. The result is stored back in 18,238.

Next the memory location variable in $51EE is decremented so that the game runs a little bit faster.

Then the score is boosted by loading 5 into B, 3 into A and calling the SCI routine given below. B carries the number of the times the digit is to be incremented and 3 tells the routine which digit to increment. So here the score is boosted by 5ØØ.

The processor then makes the long branch back to the NLV routine which will put the next level up on the screen.

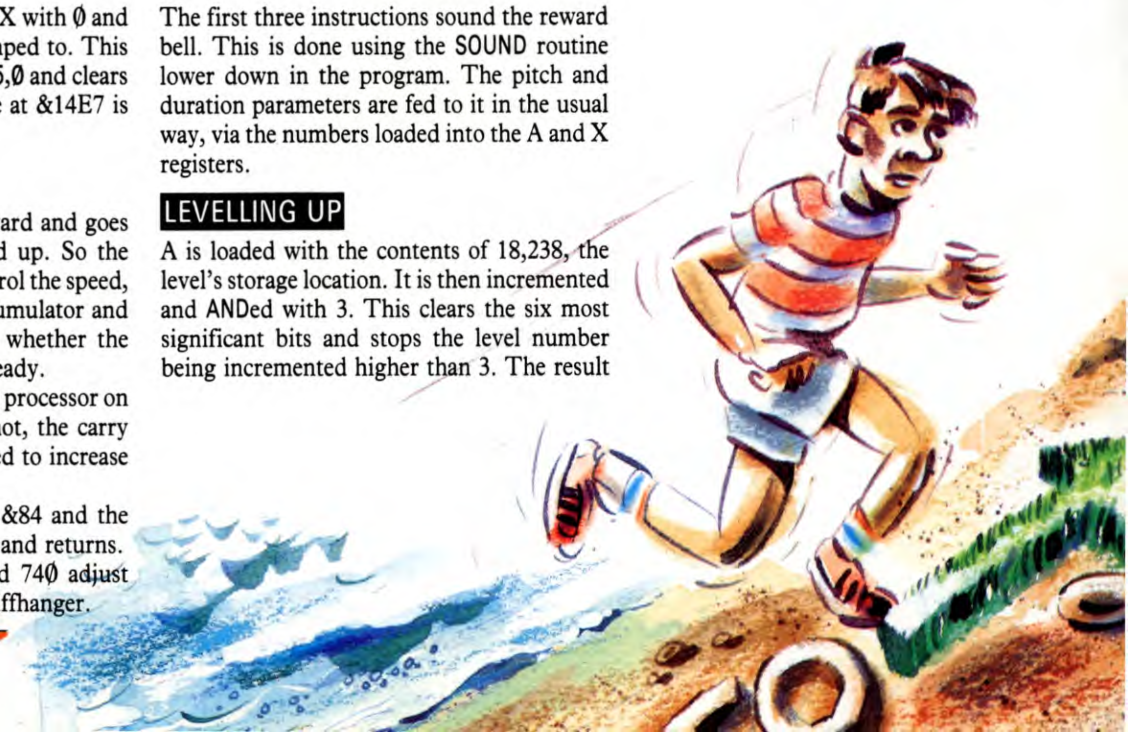
KNOW THE SCORE

For the purposes of the next routine the contents of the A and B registers have to be swapped round. This is done by an EXG—or EXchanGe—instruction. Then X is loaded with 18,24Ø, the start address of the score data.

ABX adds the contents of B—which is the number of the digit to be incremented—to X. In other words, it shifts the pointer in X along from the beginning of the score data to the location of the actual digit that you want to increment.

Then the contents of A—the number of times that digit is to be incremented—and X—the memory location of that digit—are pushed onto the hardware stack. And the processor jumps to the SDI subroutine which does the incrementing. The contents of A and X are then pulled back off the stack.

A is then decremented and the BNE SCT instruction sends the processor round the loop again, if A has not been decremented to zero. So this digit incrementing routine is



executed A times, incrementing the appropriate digit once each time it goes round.

When A has been counted down to zero and the score has been worked out, the processor drops out of the loop and jumps to the PRSC routine. This is the one that prints the score up on the screen.

And when it returns from doing that, the processor hits another RTS and returns to the main routine where this one was called.

DABBLING WITH DIGITS

A is loaded with the contents of the memory location pointed to by X. This is the actual number comprising the appropriate digit of the score. It is then incremented!

The result is compared to 1Ø. If it is not 1Ø, the BNE instruction skips the processor forward to the label SNO where the incremented digit in A is stored back in the location pointed to by X and the processor returns.

But if it is, the digit pointed to by X is cleared—that is, it is set back to zero. Then X is decremented. This moves the X pointer on to the location containing the next digit to the left.

BRA SDI then sends the processor back round the digit incrementing loop. This increments the next digit and checks to see whether the next digit has overflowed.

Eventually, the processor will find a digit that does not overflow and store its new value back in the location pointed to by X.

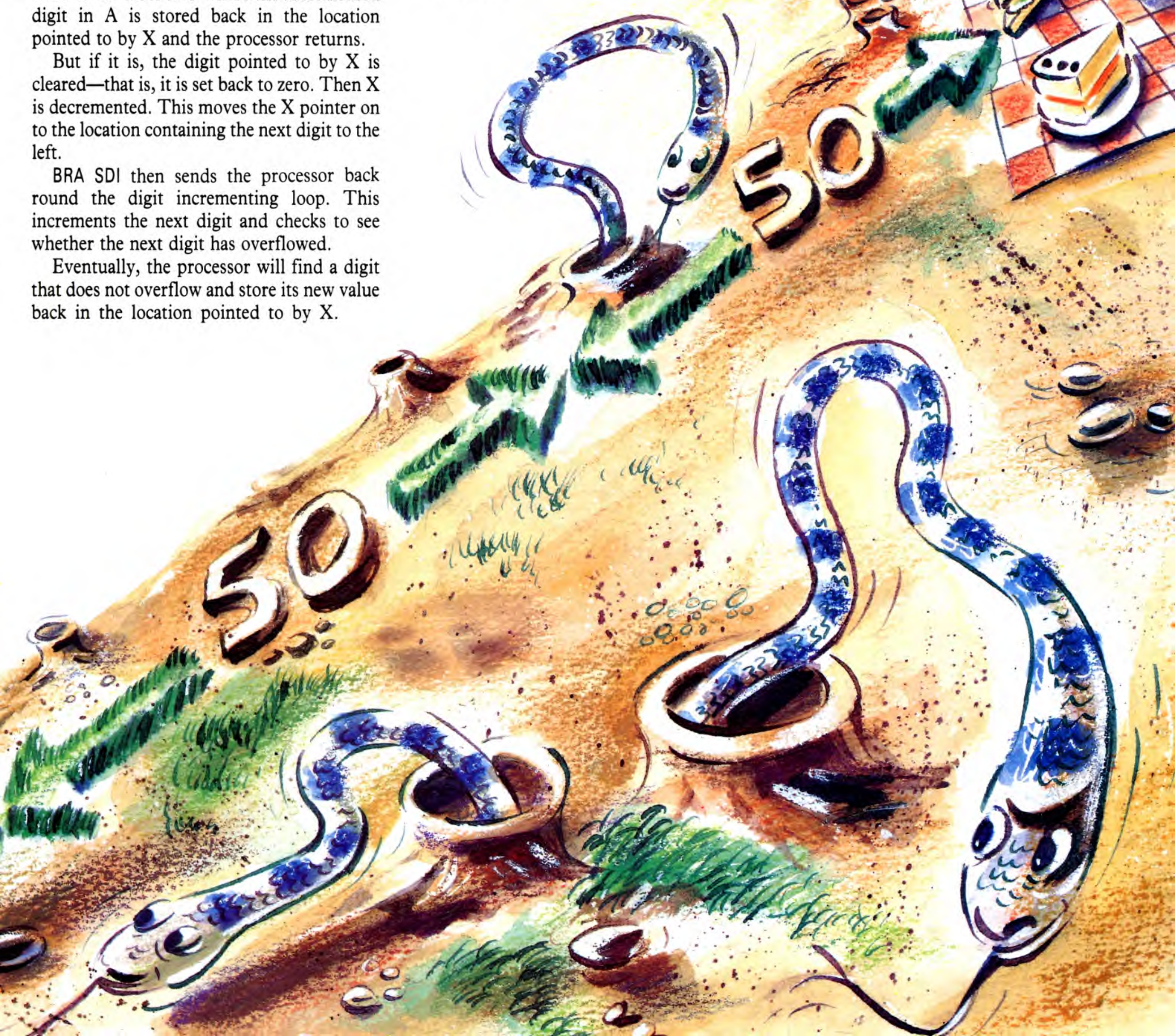
You'll note that when the processor gets back to the score routine where it was called, X is restored by pulling it off the stack.

The SOUND routine here works exactly the same way as the routine used to play Greensleeves (see page 972). But this only plays one note at a time and does not use the user stack as its source of music data.

After the SOUND routine there is another little routine called CLICK. This does Willie's walking sound by loading the sound parameters into X and A and calling SOUND.

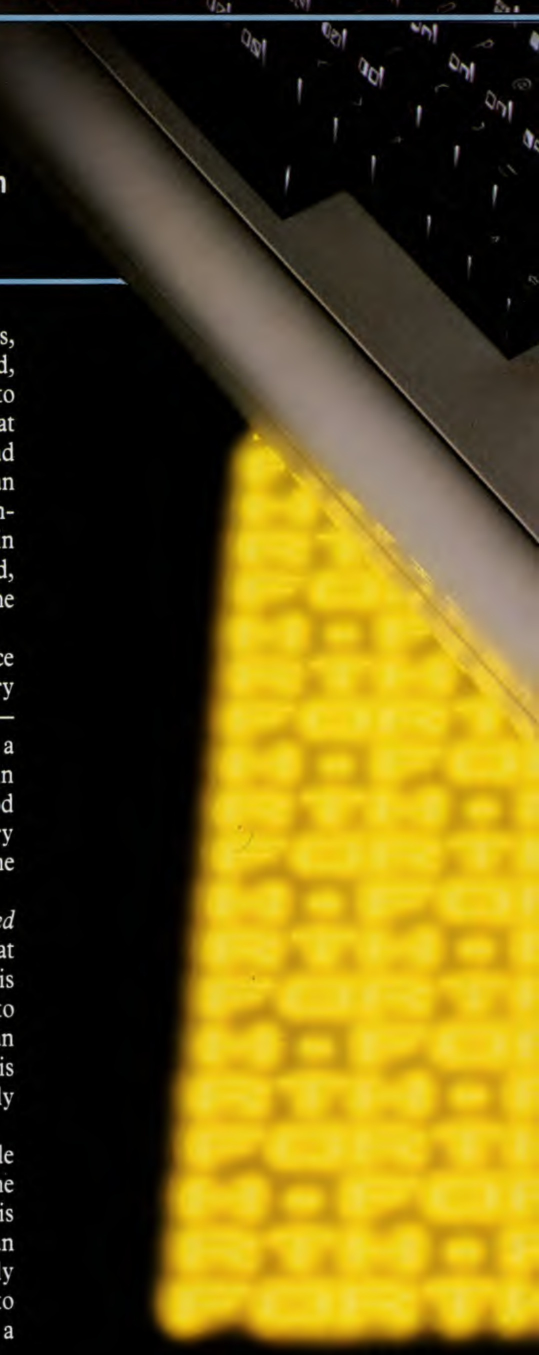
You can test the program using the M and N keys and this short BASIC program:

```
1Ø POKE 3ØØØ,S7
2Ø EXEC 19426
3Ø EXEC 199Ø2
4Ø GOTO 3Ø
```



AND SO... FORTH

Originally invented for functional programming, FORTH, with its totally transportable programs and speed, is now being used on a much wider scale on many machines



FORTH is a very efficient high-level programming language and operating system. Its speed is closer to machine code than to BASIC—but unlike machine code, it is very easy to learn. And one of the supreme advantages of FORTH programs is that they are almost completely transportable from one computer to another. This applies not just between small home micros, either—the machine's size or type has very little to do with it. Transportability like this is a very rare attribute in an industry beset with the problems of incompatibility—especially within computer languages.

Several dialects and updates of FORTH exist but the core principles are the same for all. The introduction to FORTH given by this series should be easy to follow by itself, but needless to say, you will need FORTH in your computer if you want to try any of the examples.

You can buy a FORTH system for all the computers covered here. This may be supplied in the form of a tape or disk or ROM cartridge and a very effective implementation of the language may require no more than 8K. This must be loaded over the top of the resident BASIC, but program applications produced using FORTH can stand on their own.

So what is FORTH, and what use is it likely to be to you?

STRUCTURE AND EFFICIENCY

FORTH was first developed in the early 1970's by Charles H. Moore who thought his invention to be so powerful that he considered it a 'fourth generation computer language'. The computer he was working on at the time, however, permitted only five-character identifiers, so he amended the name to FORTH.

The language was originally intended for what is usually termed *functional programming*—scientific and industrial process control applications, robotics and so forth. But it can also be used for the same general purposes as any other programming language. And it's a lot faster in execution than some—typically twenty times faster than BASIC, for instance.

Like some other 'modern' languages, FORTH programs are highly structured, with a modular design, and so are very easy to get to grips with. Practically, this means that programming is very straightforward—and therefore quick, certainly much more so than working in assembly language. FORTH cannot entirely replace assembly language in situations where extreme speed is required, but it is usually easy to incorporate the necessary routines when you have to.

FORTH coding is extremely compact once compiled and in fact requires less memory than equivalent assembly language routines—even a 1K program will be capable of doing a great deal. This makes FORTH popular in situations where the program has to do a good deal of work but where there may be memory restrictions on the host computer, as on home computers, for example.

You have met the concept of a *compiled* language before in this series. This means that when a program has been entered, it is compiled, or translated, once and for all into machine code. This is in contrast to an *interpreted* language like BASIC, which is translated while the program is actually running—a much slower process.

The process in FORTH is actually a little more complicated than either of these and one of the interesting things about the language is that it functions both as a compiler and as an interpreter, though normally the latter. Only when new words are being added to FORTH's vocabulary does it act as a compiler.

During its interpreter mode, FORTH attempts to match definitions to the program instructions which have been entered and have to be executed. But source code for a program doesn't normally come in the form of single line entries. Instead, *screens* are used to form the *input stream*—which is literally a stream of incoming data and instructions.

This is invariably read in from a storage device, usually a disk although it is possible from tape. A screen consists of a block of 1024 bytes of storage data space which corresponds to the display space available on a typical screen display. This is the only connection there ought to be between the use of the term screen in FORTH, and its normal application.

Now the FORTH interpreter can look at the full block (screenful) of data and take this as the input stream. Several such screens may be needed for a program and it is possible to chain them so they are self-loading.

THE WAY OF THE WORD

A FORTH program is composed of a series of functions (and operators) which are held in a sort of reference bank called a *dictionary*. You can use this dictionary of *words*—the FORTH equivalent of a command—to create new words of such complexity that a single one

may act like a complete program. And you can combine this new word with others to create still more powerful command words. This is similar to the way in which you have seen LOGO and LISP working, where the simple, inbuilt functions are chained together to make more and more complex procedures.

A word *definition* has two parts—the first is the *header* made up of the name which has been given to the new word. The second part is the *body* which can consist of words and/or numbers and/or operators. The whole definition is entered using *line input* between a colon and semi-colon.

Line input is the term used to describe the entry of any group of words and figures prior to pressing RETURN or ENTER. FORTH responds to such an input with the abbreviation OK if the entry is accepted, and ? if there has been an error of some kind. Additional system messages may also be displayed in the second case.

So line input of a definition takes the form:

```
: newword oldword operator;
```

When newword is subsequently executed (carried out) it'll achieve exactly the same result as the oldword if this had been operated upon instead.

Now suppose you wanted to repeat the action achieved by newword several times. Obviously it would be rather tedious and wasteful of space to reuse newword on each occasion. So why not set up a new definition? Here's how it would look:

```
: rerun newword newword newword;
```

Now you need only to call up rerun to repeat newword three times. And of course, each time rerun is called into play this calls the original based on oldword and its operator. You can see each new definition automatically embodies all of the old ones.

The word itself can be composed of almost any combination of characters available on the computer except control and graphics symbols. Obviously, there's a good deal of sense in giving meaningful names to these definitions. You need to be careful how you use spaces, however. Spaces are very important in word definitions and in the general command structure of FORTH, because they are used to mark the end of the word itself.

Let's look at a 'real' definition using one of the resident words ." which is called the dot-quote:

```
:GREET1. "THANK YOU VERY MUCH";
:GREET2 GREET1. "I AM FEELING";
```

There are no prizes for guessing what the screen will display when GREET2 RETURN is entered...

LOOKING IN THE DICTIONARY

Using these principles, a whole chain of father-begat-son definitions can be created, but their ancestry must trace right back to the core dictionary. There are many useful words contained there and it's well worthwhile getting to grips with functions and meanings of each word before you get seriously into FORTH, for there's often the very real danger of attempting to re-invent the wheel!

The actual number of words held in the dictionary—the so called *subset*—depends on the FORTH implementation you are using but the full list can be inspected at will (using the command word VLIST. A typical core dictionary contains some 200 or 300 words. Their definition closely follows the standards laid down by the various FORTH bodies and this is one of the things that ensures the very high degree of program portability possible with this language.

The dictionary may actually consist of more than one *vocabulary*—the primary one is called FORTH and this is the word to key (and execute) by pressing RETURN or ENTER when you want to return to what is called the *context vocabulary*.

All vocabularies link back to the FORTH vocabulary and this helps to justify their existence. When commands are entered, FORTH first looks through whatever happens to be the current vocabulary and then refers back to the FORTH vocabulary in order to find a match for the word under execution. As soon as it finds such a match, the corresponding definition is carried out. Putting the definition of rarely used words into special vocabularies speeds up the search time when in the FORTH vocabulary itself.

THE STACK

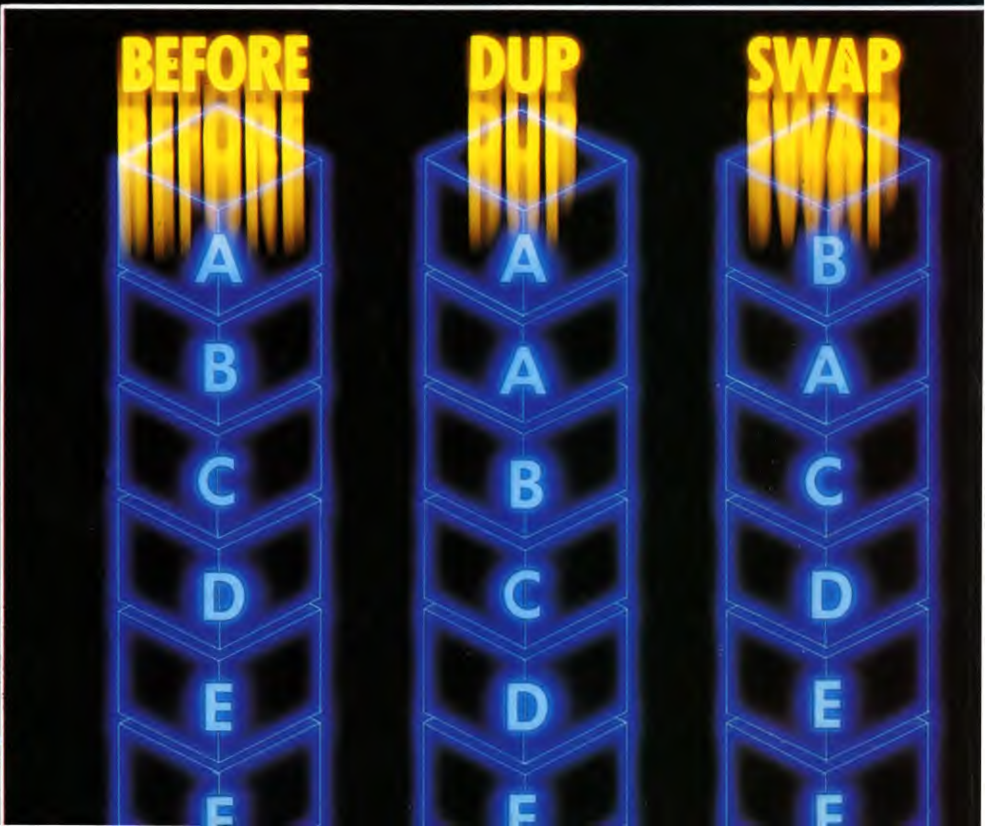
The working of FORTH—indeed its whole structure—is built around what is termed the *stack*. This is like the stack used in assembly language programming and is used both to hold and to transfer data items—numbers—for use in various parts of a program.

By its very nature, information used in the execution of a program sequence need only be of a temporary nature and this is why the concept of a stack is so very important.

A stack works on the 'last in first out' (LIFO) principle, also referred to as 'push-on pop-off'. It is perhaps easiest to think of this in terms of an analogy. Imagine a stack of dining plates—ideally on one of those sprung canteen plate dispensers sunk in a way that leaves the topmost plate always level with the surface. As a plate is taken off, so the one below is pushed to the top—and as a plate is added, this pushes down the rest.

These plates can be likened to the way data items are treated. When added to the stack, a data item is *pushed on*, and may be 'buried' by other data items subsequently pushed onto the stack. The first item can be removed—or 'popped off' easily enough—it's on top. Anything below it has to have what's above it on the stack removed before it can be accessed.

All you have to do to put a number on the stack is to type it in and press RETURN. It can



be recalled by using FORTH's *dot* command (a full stop .) which prints the topmost number of the stack if there is one, otherwise zero is displayed.

Several dots may be used to call out more than a single numeral. So if you were to key in 5 4 3 2 RETURN (remembering the spaces) and follow this with four dots, you would have the numbers 2 3 4 5 displayed before the OK prompt. Enter another dot and you will get zero displayed because the stack is now empty. Follow this with yet another dot and the error message ? EMPTY STACK or ?STACK EMPTY is printed out.

The stack enables various parts of a FORTH program to communicate with each other. Various low-level routines can access information that has been placed on the stack, remove or modify this as necessary and then return it back to the stack for use by another part of the program.

To give a simple example, think of the addition of 5 and 7. First 5 and then 7 are placed on the stack as separate data items. Next the program needs some sort of instruction to add these two numbers—a function already defined in the dictionary by the word + (this is a word, and *not* a symbol in FORTH).

Thus the key sequence is:

```
5 7 +
```

If you were to press RETURN now, the line would terminate with OK to indicate that these data items had been placed on the stack. But the presence of + in the line entry has forced execution of at least part of a procedure, for this is called *plus* (perhaps not surprisingly!) and the word + is defined as 'leave the sum of n1 and n2 (on the stack)'.

Following this instruction, the sum of n1 and n2 is passed to the stack and can be revealed using the dot command followed by RETURN to print the last entry on the stack. The resulting line display would be:

```
5 7 + . 12 OK
```

You could just as well have entered 5 7 + and . each as separate line entries to give the message 12 OK.

It is important to notice the form in which the sum is entered—the mathematical operator is entered *after* the numbers to which it refers. This is familiar to anyone who has used an ordinary calculator. Both these and FORTH operate on the concept of what is termed *reverse Polish notation* (RPN), otherwise called *postfix notation* (PFN). This is unlike *infix* notation, used in conventional writing arithmetic or the *prefix* (Polish) notation used in LISP and LOGO, for example.

FORTH has to use PFN so that use can be made of the stack. The 'conventional' arithmetic form 5 + 7 seems to be much more readable, but of course if a stack is being used, the operator + has nothing to work on because the second number is not present on the stack when the operator is called.

For those unfamiliar with PFN, even simple arithmetic may become a daunting task. But it is made simpler by remembering the LIFO principle of the stack.

The first rather obvious rule is that all the values have to be on the stack before you can actually do anything! The operators you can use follow conventional practice and several operations may be performed in a single input entry. You can control when operations are done by specifying the order of the operators. For example:

```
7 9 3 + * . 84 OK
```

is equivalent to (3 + 9) *7 and:

```
7 9 3 * + . 34 OK
```

is equivalent to 3*9 + 7.

STACK MANIPULATION

Simply calling numbers off the top of the stack isn't really too useful by itself, and this is really the point where you need to consider the FORTH words which are available for duplicating, changing the position or removing stack entries.

The stack manipulation commands include the following:

Word/Purpose	*Before/after Example*
DUP duplicates the topmost stack value	n1 ——— n1 n1 69 DUP . . 69 69 OK
DROP removes and discards the topmost value	n1 ——— 64 15 DROP . . 64 Ø STACK EMPTY
SWAP exchange the two topmost stack values	n1 n2 ——— n2 n1 17 9 SWAP . . 17 9 OK
OVER puts a copy of the second item on the stack	n1 n2 ——— n1 n2 n1 56 13 OVER . . . 56 13 56 OK
ROT rotates the top three times on the stack	n1 n2 n3 ——— n2 n3 n1 3 8 9 ROT . . . 8 9 3 OK

Some interesting points appear here. The first is the 'before and after' arrangement of the display explaining the effect of a particular word. Every FORTH glossary gives this as:

before ——— after

The dashes suggest the presence and action of the word used on things—numbers—that should be on the stack before execution. This is called stack notation.

The second point—which leads on to complex program constructions in FORTH—is that it should be quite clear that the various manipulations enable you to circumvent many of the restrictions imposed by the LIFO nature of the stack. The next article will show how a FORTH program evolves.

ESCAPE: THE ADVENTURE GOES ON

Here is the third part of *INPUT*'s adventure game. Remember, there are no clues in this series of articles, so don't be mislead by the illustrations.

When you have finished adding the latest lines of programming, do not forget to SAVE the program ready for next time.

```
200 RESTORE 4020: FOR Z=1 TO 21
210 READ K(Z),F(Z): LET NN=Z*2+124: GOSUB 4500: LET O$(Z)=S$: LET NN=Z*2+125: GOSUB 4500: LET E$(Z)=S$
230 NEXT Z
240 FOR Z=1 TO 32
250 READ R(Z): LET NN=167+Z: GOSUB 4500: LET R$(Z)=S$
260 NEXT Z
530 INPUT INVERSE 1;"WHAT NOW?",LINEI$
535 IF I$="" THEN GOTO 530
540 IF I$=U$ AND TT=Ø OR I$=J$ AND II=Ø THEN GOSUB 3Ø4Ø: GOTO 27Ø
550 LET X$=I$: LET Y$=CHR$ 32: GOSUB 5ØØØ
560 IF IN=Ø THEN LET V$=I$: GOTO 58Ø
570 LET V$=I$( TO IN-1)
580 LET T$=I$(IN+1 TO )
590 IF V$="GO" THEN LET V$=T$
640 LET I=Ø
650 FOR Z=1 TO 32
660 LET X$=R$(Z): LET Y$=V$: GOSUB 5ØØØ: IF IN=1 THEN LET I=R(Z)
670 NEXT Z
680 IF I<1 THEN PRINT "I DON'T KNOW HOW TO□";I$: GOTO 53Ø
690 IF E$(L,1)<>CHR$ 32 AND I<>9 AND I<>1Ø AND I<>5 AND I<>12 AND I<>8 AND F=1 THEN PRINT "THE□";E$(L),"WON'T LET YOU.": PAUSE 1ØØ: GOTO 27Ø
1760 REM PROC A
1770 CLS
1780 PRINT AT 11,6;: LET NN=65: GOSUB 396Ø
1800 LET NN=33: GOSUB 396Ø
1810 PAUSE 75Ø
1820 GOSUB 184Ø
1830 RETURN
1840 REM PROC B
1850 CLS
1860 PRINT FLASH 1;AT 11,8; "YOU'RE DEAD!"
1880 STOP
1890 CLS
1900 IF K(3)<>-1 THEN LET NN=66: GOSUB 396Ø: LET NN=67: GOSUB 396Ø: PAUSE Ø: RETURN
1910 IF K(2)<>-1 THEN LET NN=68: GOSUB 396Ø: LET NN=67: GOSUB 396Ø: PAUSE Ø: LET L=L-6: RETURN
1920 PRINT FLASH 1;AT 11,1Ø; "WELL DONE!"
1940 PRINT FLASH 1;AT 16,1Ø; "YOU'VE WON"
1960 STOP
1970 REM PROC C
1980 LET PQ=Ø: FOR Z=1 TO 21
1990 LET X$=O$(Z): LET Y$=T$: GOSUB 5ØØØ: IF IN>Ø THEN LET PQ=Z
2000 NEXT Z
2010 IF PQ=Ø THEN PRINT "I DON'T UNDERSTAND□";T$;".": GOTO 2Ø7Ø
2020 IF K(PQ)=-1 THEN PRINT "YOU'VE ALREADY GOT IT!": GOTO 2Ø7Ø
2030 IF K(PQ)<>L THEN PRINT "THE□"; T$;"□IS NOT HERE!": GOTO 2Ø7Ø
2040 IF PP>3 THEN PRINT "YOU CAN'T CARRY ANY MORE.": GOTO 2Ø7Ø
2050 LET K(PQ)=-1: LET PP=PP+1: PRINT "OKAY—YOU'VE GOT IT."
2060 LET NN=166: GOSUB 45ØØ: IF T$=S$ THEN LET XX=XX+6ØØ
2070 PAUSE 1ØØ
2080 RETURN
2090 REM PROC D
2100 DIM G$(17): LET NN=153: GOSUB 45ØØ: LET G$=S$: DIM B$(17): LET NN=155: GOSUB 45ØØ: LET B$=S$: IF E$(L)=G$ OR E$(L)=B$ THEN LET NN=69: GOSUB 396Ø: PRINT E$(L): LET NN=34: GOSUB 396Ø: PAUSE 25Ø: STOP
2110 IF E$(L,1)=CHR$ 32 THEN LET NN=35: GOSUB 396Ø: PAUSE 3ØØ: GOTO 244Ø
2120 DIM G$(17): LET NN=129: GOSUB 45ØØ: LET G$=S$: IF G$=E$(L) THEN LET NN=36: GOSUB 396Ø: PAUSE 35Ø: GOTO 244Ø
2130 LET C$="": LET H$=""
2140 LET WW=1: LET AA=Ø
2150 IF K(2Ø)=-1 THEN LET WW=WW+2
2160 IF K(9)=-1 THEN LET WW=WW+3
2170 IF K(15)=-1 THEN LET AA=AA+ .5
2180 IF K(14)=-1 THEN LET AA=AA+ .5
2190 IF K(1Ø)=-1 THEN LET WW=WW+1
2200 IF AA=1 THEN LET WW=WW+3
2210 IF WW=1 AND C$="" THEN INPUT "DO YOU WANT TO FIGHT WITH BARE HANDS? (Y/N)",H$
2220 DIM G$(17): LET NN=7Ø: GOSUB 45ØØ: LET G$=S$: IF E$(L)=G$ AND H$="Y" THEN LET NN=37: GOSUB 396Ø: PAUSE 25Ø: GOSUB 184Ø
2230 DIM G$(17): LET NN=135: GOSUB 45ØØ: LET G$=S$: IF H$="Y" AND F(L)>1 AND E$(L)<>G$ THEN PRINT "YOU CAN'T FIGHT THE□";E$(L),"WITH BARE HANDS!": PAUSE 15Ø: GOTO 244Ø
2240 IF H$<>"Y" AND WW=1 THEN GOTO 244Ø
2250 LET EE=INT (RND*6)+1: CLS
2260 FOR Z=Ø TO 21
2270 PRINT AT Z,Ø;
2280 NEXT Z
2290 PRINT FLASH 1;AT 1Ø,12;"FIGHTING!"
2310 PAUSE 1ØØ
2320 IF WW>F(L) AND EE>2 THEN LET V=V-2: GOTO 236Ø
2330 IF WW>F(L) AND EE<=2 THEN LET V=V-1: GOTO 238Ø
2340 IF WW<=F(L) AND EE>=4 THEN LET V=V-3: GOTO 236Ø
2350 IF WW<=F(L) AND EE<4 THEN LET V=V-3: GOTO 238Ø
2360 IF V<1 THEN GOSUB 184Ø
2370 PRINT '"YOU ARE WOUNDED."'"YOUR VITALITY IS ";V: GOTO 241Ø
2380 IF V<1 THEN GOSUB 184Ø
2390 PRINT '"YOU HAVE WON THE BATTLE."'"YOUR VITALITY IS□";V
2400 LET E$(L)="": PAUSE 15Ø: RETURN
2410 LET LL=INT (RND*21)+1: IF K(LL)=-1 THEN PRINT "YOU HAVE DROPPED THE",O$(LL): LET K(LL)=L: LET PP=PP-1
2420 INPUT "DO YOU WANT TO CONTINUE THE□□□□□FIGHT? (Y/N)", LINE C$
2430 IF C$="Y" THEN GOTO 214Ø
2440 CLS : RETURN
```

Continue entering Escape, *INPUT*'s new adventure game. LOAD in the existing program and add these lines. The program cannot be RUN until it is completed

```
2450 REM PROC E
2460 CLS
2470 PRINT "YOU HAVE COLLECTED: —":
   LET PP=Ø
2480 FOR Z=1 TO 21
2490 IF K(Z)=−1 THEN PRINT
   "THE□";O$(Z): LET PP=PP+1
2500 NEXT Z
2510 IF PP=Ø THEN PRINT "NOTHING"
2520 PRINT "YOUR VITALITY IS□";V
2530 PAUSE 250
2540 RETURN
2550 REM PROC F
2560 CLS
2570 LET NN=71: GOSUB 3960
2580 PAUSE 50
2590 LET J=INT (RND*6)+1
2600 IF TT<1 AND II<1 THEN LET
   NN=72: GOSUB 3960: GOTO 2720
2610 IF TT=1 AND II=1 AND J>3 THEN
   GOTO 2680
2620 IF TT<1 AND II=1 THEN GOTO 2680
2630 LET NN=38: GOSUB 3960
2640 PAUSE 300: LET U$=""
2650 FOR Z=Ø TO 5: LET PQ=INT (RND*26)
   +97: LET U$=CHR$ (PQ)+U$: NEXT Z
2660 LET NN=73 GOSUB 3960: PRINT U$:
   LET TT=Ø
2670 PAUSE 150: GOTO 2730
2680 LET NN=39: GOSUB 3960
2690 PAUSE 250: LET J$=""
2700 FOR Z=Ø TO 5: LET PQ=INT
   (RND*26)+97: LET J$=CHR$ (PQ)+J$:
   NEXT Z
2710 LET NN=73: GOSUB 3960: PRINT J$:
   LET II=Ø
2720 PAUSE 150
2730 CLS : RETURN
2740 REM PROC G
2750 IF E$(L,1)<>CHR$ 32 THEN RETURN
2760 CLS : LET NN=74: GOSUB 3960
2770 LET E$(L)=M$: LET F(L)=10
2780 LET N=Ø: LET S=Ø: LET E=Ø: LET
   W=Ø: LET U=Ø: LET D=Ø: LET F=1
2790 RETURN
2800 REM PROC H
2810 IF E$(L,1)=CHR$ 32 THEN LET
   NN=75: GOSUB 3960: PAUSE 100: GOTO
   2980
2820 LET NN=153: GOSUB 4500: DIM
   G$(17): LET G$=S$: IF E$(L)=G$ THEN
```

```
   GOTO 2826
2822 LET NN=155: GOSUB 4500: DIM
   G$(17): LET G$=S$: IF E$(L)=G$ THEN
   GOTO 2826
2825 GOTO 2830
2826 PRINT "NO DEAL!": LET NN=34:
   GOSUB 3960: PAUSE 250: STOP
3960 REM DECODE & PRINT STRING
3970 LET Z(1)=A(NN): LET XXX=USR
   65067: PRINT "Z$: RETURN
4020 DATA 1,0,2,0,3,0,0,0,0,4,8,0,KK,0,8,0,
   9,0,10,0,11,2,12,0,0,0,14,4
4030 DATA 15,6,0,0,17,0,0,0,19,0,20,1,21,0
4040 DATA 8,5,5,4,8,9,10,9,10,11
4050 DATA 2,2,12,3,3,1,1,1,1,1,1,1
4060 DATA 6,7,12,12,1,1,1,1,1,1
4500 REM DECODE STRING INTO S$
4510 LET Z(1)=A(NN): LET XXX=USR
   65067: LET X$=Z$: LET Y$+CHR$9
4520 GOSUB 5000: LET S$=Z$ (TO IN-1)
4530 RETURN
5000 REM INSTR ROUTINE
5010 LET IN=0: IF LEN Y$>LEN X$ THEN
   RETURN
5020 FOR Z=1 TO (LEN X$-LEN Y$+1)
5030 IF Y$=X$(K TO K+LEN Y$-1) THEN
   LET IN=Z: LET Z=(LEN X$-LEN Y$-1)
5040 RETURN
```

```
285 E$(NN)=Z$:IF E$(NN)="□"THEN
   E$(NN)=""
385 ONL-9GOSUB1190,1260,1240,1330,
   1470,1640,1790,1340,1060,1310,1360,
   1140,2080
665 IF (I-1)<1 THEN V$="":GOTO 670
666 V$=LEFT$(I$,I-1)
1670 IF INT(RND(1)*18)<4 THEN F=0:
   TX=63:GOSUB 9900
1680 RETURN
1690 IF INT(RND(1)*18)+1=3 AND
   DW=1 THEN 3100
1700 PRINT "♥"
1710 N=0:S=1:E=0:W=1:
   U=0:D=0
1720 PRINT:TX=25:
   GOSUB 9900
1730 IF K(17)=-1 THEN TX=26:
   GOSUB 9900:D=1
1740 RETURN
1750 PRINT"♥"
1760 N=1:S=0:E=0:W=0:
   U=0:D=0
1770 PRINT:TX=27:
   GOSUB 9900
1780 RETURN
1790 PRINT"♥"
1800 N=0:S=1:E=0:W=0:
   U=1:D=0
1810 PRINT:TX=12:
   GOSUB 9900
1820 IF K(7)=-1 THEN TX=64:
   GOSUB 9900:D=1
1830 RETURN
1840 IF INT(RND(1)*18)=1 THEN GOSUB
   2860
1850 PRINT "♥":N=1:S=1:E=1:W=0:
   U=0:D=1
1860 PRINT:TX=30:
   GOSUB 9900
1870 RETURN
1880 IF INT(RND(1)*18)+1=1 AND
   DW=1 THEN 3100
1890 PRINT "♥":N=1:S=1:E=1:W=1:
   F=0
1900 PRINT:TX=31:
   GOSUB 9900
1910 IF E$(L)<>"" THEN PRINT"[]HERE
   IS A■"E$(L)"■PASSING.":F=1
1920 RETURN
1930 PRINT "♥":N=1:S=1:E=0:W=1:
   U=0:D=1
1940 PRINT:TX=32:
   GOSUB 9900
1950 RETURN
1960 PRINT "♥■"TAB(255)TAB(168);:
   TX=65:GOSUB9900
2000 TX=33:GOSUB 9900
2010 FOR DL=1 TO 1000:IF PEEK
   (198)=64 THEN NEXT DL
2050 PRINT"♥■"TAB(255)TAB(172)
   "YOU'RE DEAD!!!":
   GOTO 10000
2080 PRINT"♥"
2090 IF K(3)<>-1 THEN PRINT:TX=66:
   GOSUB 9900:PRINT:TX=67:
   GOSUB 9900
2100 IF K(3)<>-1 THEN GET D$:IF
   D$="" THEN 2100
2110 IF K(3)<>-1 THEN RETURN
2120 IF K(2)<>-1 THEN PRINT:TX=68:
   GOSUB 9900:PRINT:TX=67:
   GOSUB 9900
2130 IF K(2)<>-1 THEN GET D$:IF
   D$="" THEN 2130
2140 IF K(2)<>-1 THEN L=L-1:
   RETURN
2150 PRINT"π"TAB(255)TAB(174)"WELL
   DONE!"
2160 PRINT TAB(13)"[]OU'VE WON."
2170 END
2180 :
2190 QQ=0
2200 FOR CC=1 TO 21
2210 FOR SC=1 TO LEN (O$(CC))-LEN
   (T$)+1
2220 IF MID$(O$(CC),SC,LEN(T$))=T$
```

```
   AND SC>Ø THEN QQ=CC:
   GOTO 224Ø
223Ø NEXT SC,CC
224Ø IF QQ=Ø THEN PRINT"□ DON'T
   UNDERSTAND "T$".":GOTO 23ØØ
225Ø IF K(QQ)=-1 THEN PRINT"□OU
   ALREADY HAVE IT!":GOTO 23ØØ
226Ø IF K(QQ)<>L THEN PRINT"□HE "
   T$" IS NOT HERE!":GOTO 23ØØ
227Ø IF PP>3 THEN PRINT"□OU CAN'T
   CARRY ANY MORE.":GOTO 23ØØ
228Ø K(QQ)=-1:PP=PP+1:PRINT
   "□KAY-YOU NOW HAVE IT."
229Ø TX=166:GOSUB 995Ø:IF T$=Z$
   THEN XX=XX+6ØØ
23ØØ GOSUB 2ØØØØ:RETURN
232Ø TX=153:GOSUB 995Ø:D1$=Z$:
   TX=155:GOSUB 995Ø
2335 IF E$(L)=D1$ OR E$(L)=Z$ THEN
   235Ø
234Ø GOTO 236Ø
235Ø TX=69:GOSUB995Ø:PRINT Z$,E$(L)
   "!":TX=34:GOSUB99ØØ:GOSUB 2ØØØØ:
   NEXT:GOTO1ØØØ
236Ø IF E$(L)="" THEN TX=35:GOSUB
   99ØØ:GOSUB 2ØØØØ:GOTO 267Ø
237Ø TX=129:GOSUB 995Ø
2375 IF E$(L)=Z$ THEN TX=36:GOSUB
```

```
   99ØØ:GOSUB 2ØØØØ:GOTO 267Ø
238Ø C$="":HS=""
239Ø WW=1:AA=Ø
24ØØ IF K(2Ø)=-1 THEN WW=
   WW+2
242Ø IF K(9)=-1 THEN WW=
   WW+3
243Ø IF K(15)=-1 THEN AA=
   AA+.5
244Ø IF K(14)=-1 THEN AA=
   AA+.5
245Ø IF K(1Ø)=-1 THEN WW=
   WW+1
246Ø IF AA=1 THEN WW=
   WW+3
247Ø IF WW=1 AND C$=""THENPRINT
   "□IGHT WITH BARE HANDS-□/
   □?":INPUT H$
248Ø TX=7Ø:GOSUB 995Ø
2485 IF E$(L)=Z$ANDH$="Y"THEN
   TX=37:GOSUB 99ØØ:GOSUB2ØØØØ:GOTO
   2Ø4Ø
249Ø TX=135:GOSUB 995Ø
2495 IF H$="Y" AND F(L)>1 AND E$(L)
   <>Z$ THEN 251Ø
25ØØ GOTO 253Ø
251Ø PRINT "□OU CAN'T FIGHT THE "
   E$(L)" WITH BARE HANDS!"
252Ø GOSUB2ØØØØ:GOTO 266Ø
253Ø IF H$<>"Y" AND WW=1 THEN
   267Ø
254Ø E=INT(RND(1)*6)+1:
   PRINT "□"
255Ø PRINT "□"TAB(255)TAB(212)
   "YOU'RE FIGHTING!"
256Ø GOSUB 2ØØØØ
257Ø IF WW>F(L) AND EE>2 THEN
   V=V-2:GOTO 26ØØ
2575 IF WW>F(L) AND EE<=2 THEN
   V=V-1:GOTO 263Ø
258Ø IF WW<=F(L)AND EE>=4 THEN
   V=V-3:GOTO 26ØØ
259Ø IF WW<F(L) AND EE<4 THEN
   V=V-3:GOTO 263Ø
26ØØ IF V<1 THEN 2Ø4Ø
261Ø PRINT:PRINT"□OU'RE WOUNDED."
262Ø PRINT "□OUR VITALITY IS ";V:
   GOTO 239Ø
263Ø IF V<1 THEN 2Ø4Ø
264Ø PRINT "□OU'VE WON THE BATTLE."
265Ø PRINT "□OUR VITALITY IS ";V
266Ø E$(L)="":GOSUB2ØØØØ:
   RETURN
268Ø LL=INT(RND(1)*21)+1
27ØØ IF K(LL)=-1 THEN PRINT"□OU'VE
   DROPPED THE "O$(LL)".":K(LL)=L:
   PP=PP-1
271Ø PRINT "□O YOU WANT TO CONTINUE
   FIGHTING-□/□?":INPUT C$
272Ø IF C$="Y" THEN 239Ø
273Ø PRINT "□":RETURN
276Ø PRINT "□□OU HAVE
   COLLECTED:-":PP=Ø
278Ø FOR CC=1 TO 21
279Ø IF K(CC)=-1 THEN PRINT "THE "
   O$(CC)",":PP=PP+1
28ØØ NEXT CC
281Ø IF PP=Ø THEN PRINT "NOTHING."
282Ø PRINT "□OU R  VITALITY IS ";V
283Ø GOTO 2ØØØØ
286Ø PRINT "□"
287Ø PRINT:TX=71:GOSUB 99ØØ
288Ø GOSUB2ØØØØ
289Ø J=INT(RND(1)*6)+1
29ØØ IF TT<1 AND II<1 THEN TX=72:
   GOSUB99ØØ:GOSUB 3Ø7Ø
291Ø IF TT=1 AND II=1 AND J>3 THEN
   299Ø
292Ø IF TT<1 AND II=1 THEN 299Ø
293Ø TX=38:GOSUB 99ØØ
294Ø GOSUB 2ØØØØ:GOSUB 2ØØØØ:
   TT$=""
295Ø FOR CC=Ø TO 5:QQ=INT(RND(1)*26)
   +65:TT$=CHR$(QQ)+TT$:NEXT CC
296Ø TX=73:GOSUB 99ØØ:PRINT
   TT$:TT=Ø
298Ø GOSUB2ØØØØ:GOTO 3Ø8Ø
299Ø TX=39:GOSUB 99ØØ
3ØØØ GOSUB 2ØØØØ:GOSUB 2ØØØØ:II$=""
3Ø2Ø FOR CC=Ø TO 5:QQ=INT(RND(1)*26)
   +65
3Ø3Ø II$=CHR$(QQ)+II$:NEXT CC
3Ø5Ø TX=73:GOSUB 99ØØ:
   PRINTII$:II=Ø
3Ø7Ø GOSUB 2ØØØØ
3Ø8Ø PRINT "□":RETURN
31ØØ PRINT "□":TX=74:
   GOSUB 99ØØ
312Ø E$(L)=JM$:F(L)=1Ø
```

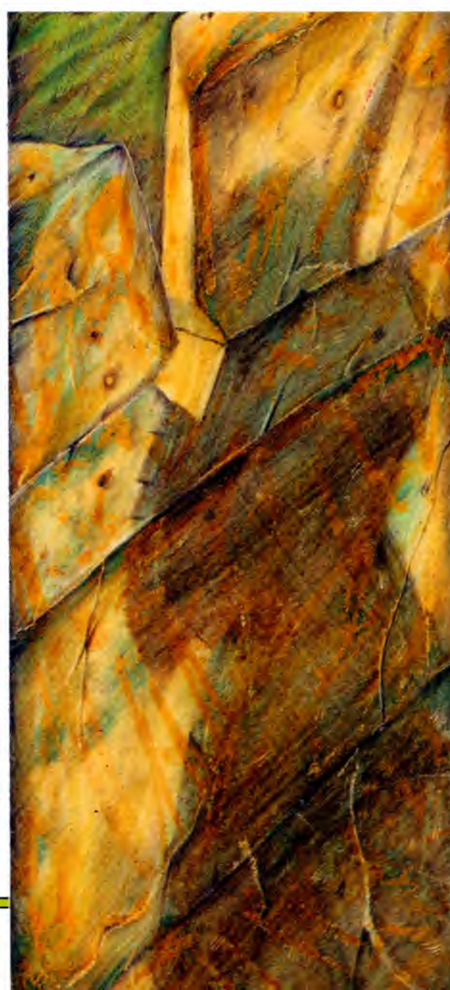
```
166Ø PRINT"FNW(3Ø)
167Ø RETURN
168Ø IF RND(18)=1 AND dw=1 THEN
   PROCG:RETURN
169Ø CLS:N=1:S=1:E=1:W=1:F=Ø
17ØØ PRINT"FNW(31)
171Ø IF E$(L)<>"" THEN PRINT"There is
   a "E$(L)" passing.":F=1
172Ø RETURN
173Ø CLS:N=1:S=1:E=Ø:W=1:U=Ø:
   D=1
174Ø PRINT"FNW(32)
175Ø RETURN
176Ø DEFPROCA
177Ø CLS
178Ø PRINTTAB(1Ø,15)CHR$(141);CHR$
   (13Ø)FNW(65)
179Ø PRINTTAB(1Ø,16)CHR$(141);CHR$
   (13Ø)FNW(65)
18ØØ PRINTFNW(33)
181Ø D=INKEY(15ØØ)
```

```
1820 PROCB
1830 ENDPROC
1840 DEFPROCB
1850 CLS
1860 PRINTTAB(1Ø,15)CHR$(141);CHR$(129)"YOU'RE DEAD!"
1870 PRINTTAB(1Ø,16)CHR$(141);CHR$(129)"YOU'RE DEAD!"
1880 END
1890 CLS
1900 IF K(3)<>−1THEN PRINT"FNW(66)"FNW(67):D$=GET$:RETURN
1910 IF K(2)<>−1THEN PRINT"FNW(68)"FNW(67):D$=GET$:L=L−6:RETURN
1920 PRINTTAB(1Ø,15)CHR$(141);CHR$(129)"WELL DONE!"
1930 PRINTTAB(1Ø,16)CHR$(141);CHR$(129)"WELL DONE!"
1940 PRINTTAB(1Ø,19)CHR$(141);CHR$(131);"You've won"
1950 PRINTTAB(1Ø,2Ø)CHR$(141);CHR$(131);"You've won"
1960 END
1970 DEFPROCC
1980 q=Ø:FOR c=1 TO 21
1990 IF INSTR(O$(c),T$)>ØTHEN q=c
2000 NEXT
2010 IF q=Ø THEN PRINT"I don't understand□"T$".":GOTO 2Ø7Ø
2020 IF K(q)=−1THEN PRINT"You've already got it!":GOTO 2Ø7Ø
2030 IF K(q)<>L THEN PRINT"The "T$"is not here!":GOTO 2Ø7Ø
2040 IF p>3 THEN PRINT"You can't carry any more.":GOTO 2Ø7Ø
2050 K(q)=−1:p=p+1:PRINT"Okay—you've got it."
2060 IF T$=FNX(FNW(166)) THEN x=x+6ØØ
2070 D=INKEY(25Ø)
2080 ENDPROC
2090 DEFPROCD
2100 IF E$(L)=FNX(FNW(153)) OR E$(L)=FNX(FNW(155))THEN PRINTFNW(69)E$(L)"!":PRINTFNW(34):D=INKEY(5ØØ):END
2110 IF E$(L)="" THEN PRINTFNW(35):d=INKEY(75Ø):GOTO 244Ø
2120 IF E$(L)=FNX(FNW(129)) THEN PRINTFNW(36):d=INKEY(7ØØ):GOTO 244Ø
2130 C$="":H$=""
2140 w=1:a=Ø
2150 IFK(2Ø)=−1 THEN w=w+2
2160 IFK(9)=−1 THEN w=w+3
2170 IFK(15)=−1 THEN a=a+.5
2180 IFK(14)=−1 THEN a=a+.5
2190 IFK(1Ø)=−1 THEN w=w+1
2200 IF a=1 THEN w=w+3
2210 IF w=1 AND C$="" THEN INPUT"You want to fight with bare hands?(y/n)"H$
2220 IFE$(L)=FNX(FNW(7Ø)) AND H$="y" THEN PRINTFNW(37):d=INKEY(5ØØ): PROCB
2230 IF H$="y" AND f(L)>1 AND E$(L)<>FNX(FNW(135)) THEN PRINT"You can't fight the □"E$(L) "□with bare hands!":D=INKEY(25Ø):GOTO 244Ø
2240 IF H$<>"y" AND w=1 THEN 244Ø
2250 e=RND(6):CLS
2260 FOR d=1 TO 2Ø
2270 PRINTTAB(Ø,d)CHR$(131);CHR$(157)
2280 NEXT
2290 PRINTTAB(1Ø,1Ø)CHR$(141);CHR$(132);CHR$(136)"Fighting!"
2300 PRINTTAB(1Ø,11)CHR$(141);CHR$(132);CHR$(136)"Fighting!"
2310 d=INKEY(2ØØ)
2320 IF w>f(L)AND e>2 THEN V=V−2:GOTO 236Ø
2330 IF w>f(L)AND e<=2 THEN V=V−1:GOTO 238Ø
2340 IF w<=f(L)AND e>=4 THEN V=V−3:GOTO 236Ø
2350 IF w<=f(L)AND e<4 THEN V=V−3:GOTO 238Ø
2360 IF V<1 THEN PROCB
2370 PRINT""You are wounded."'"Your vitality is□"; V:GOTO 241Ø
2380 IF V<1 THEN PROCB
2390 PRINT""You have won the battle."'"Your vitality is□";V
2400 E$(L)="":D=INKEY(25Ø):ENDPROC
2410 l=RND(21):IF K(l)=−1 THEN PRINT"You have dropped the□"O$(l)".":K(l)=L:p=p−1
2420 INPUT"Do you want to continue the fight?(y/n)"C$
2430 IF C$="y" THEN214Ø
2440 CLS:ENDPROC
2450 DEFPROCE
2460 CLS
2470 PRINT"You have collected:−":p=Ø
2480 FOR c=1 TO 21
2490 IF K(c)=−1 THEN PRINT"the "O$(c)",":p=p+1
2500 NEXT
2510 IF p=Ø THEN PRINT"nothing"
2520 PRINT"Your vitality is□";V
2530 D=INKEY(5ØØ)
2540 ENDPROC
2550 DEFPROCF
2560 CLS
2570 PRINT"FNW(71)
2580 d=INKEY(1ØØ)
2590 J=RND(6)
2600 IF t<1 AND i<1 THEN PRINT FNW(72):GOTO 272Ø
2610 IF t=1 AND i=1 AND J>3 THEN 268Ø
2620 IF t<1 AND i=1 THEN 268Ø
2630 PRINTFNW(38)
2640 d=INKEY(7ØØ):t$=""
2650 FOR c=Ø TO 5:q=RND(26)+96:t$=CHR$(q)+t$:NEXT
2660 PRINT FNW(73)'t$:t=Ø
2670 D=INKEY(3ØØ):GOTO 273Ø
2680 PRINTFNW(39)
2690 d=INKEY(5ØØ):i$=""
2700 FOR c=Ø TO 5:q=RND(26)+96:i$=CHR$(q)+i$:NEXT
2710 PRINTFNW(73)'i$:i=Ø
2720 D=INKEY(3ØØ)
2730 CLS:ENDPROC
2740 DEFPROCG
2750 IF E$(L)<>"" THEN ENDPROC
2760 CLS:PRINT"FNW(74)
2770E$(L)=JM$:f(L)=1Ø
2780 N=Ø:S=Ø:E=Ø:W=Ø:U=Ø:D=Ø:F=1
2790 ENDPROC
2800 DEFPROCH
2810 IF E$(L)="" THEN PRINTFNW(75):D=INKEY(2ØØ):GOTO 298Ø
2820 E$(L)=FNX(FNW(153)) OR E$(L)=FNX(FNW(155)) THENPRINT"No Deal!"'FNW(34):D=INKEY(5ØØ):END
2830 PRINTTAB(Ø,2Ø);:INPUT"What are you prepared to offer?"offer$:value=Ø
2840 IFK(21)=−1AND offer$=FNX(FNW(166))THEN 289Ø
2850 IF offer$=FNX(FNW(166))THEN PRINT FNW(76):D=INKEY(3ØØ):ENDPROC
2860 IF offer$=FNX(FNW(126)) OR offer$=FNX(FNW(77)) AND K(1)=−1 AND E$(L)=FNX(FNW(7Ø)) THEN PRINT"It's a deal":D=INKEY(25Ø):E$(L)="":K(1)=Ø:ENDPROC
2870 IF offer$=FNX(FNW(126)) AND E$(L)=FNX(FNW(7Ø)) THEN PRINTFNW(78):D=INKEY(25Ø):GOTO53Ø
2880 IF offer$<>FNX(FNW(166))THEN PRINTFNW(79)offer$"?":D=INKEY(35Ø):GOTO 298Ø
2890 PRINT"You have ";x"□gold □sovereigns."'FNW(8Ø);:INPUToffer
2900 IF offer>x THEN PRINTFNW(81):GOTO 289Ø
2910 CLS:PRINT"FNW(82):D=INKEY(25Ø)
2920 price=RND(12)*5Ø
2930 IFprice>offer□CLS:PRINT"FNW(83)'FNW(84);:INPUT""inc$
2940 IF offer>=price□THEN299Ø
2950 IF inc$="y" THEN 289Ø
2960 IFE$(L)<>FNX(FNW(129))THENPRINTFNW(85):D=INKEY(25Ø):PROCD:ENDPROC
2970 PRINTFNW(86):D=INKEY(3ØØ):PRINT"You surrender.":D=INKEY(1ØØ):END
2980 CLS:ENDPROC
2990 PRINT "Okay — It's a deal": E$(L)="": x=x−offer
3000 IF x=Ø THENK(21)=21
3010 IF offer<>Ø THEN PRINT"You've lost □";offer; "gold sovereigns."
```

```
3020 D=INKEY(25Ø)
3030 ENDPROC
3040 DEFPROCI
3050 CLS
3060 PRINT''FNW(87)
3070 IF I$=i$ ANDi=Ø THEN 3250
3080 IF I$=t$ AND t=Ø THEN
   PRINT':INPUT"Where do you wish to
   go",d$:t=-1
3090 IF d$=FNX(FNW(88)) OR d$=FNX
   (FNW(89)) THEN L=19:ENDPROC
3100 IF d$=FNX(FNW(9Ø))OR d$=FNX
   (FNW(91))THEN L=1:ENDPROC
3110 IF d$=FNX(FNW(92))THEN L=8:
   ENDPROC
3120 IF d$=FNX(FNW(93)) ORd$=FNX
   (FNW(94)) THEN L=2:ENDPROC
3130 IF d$=FNX(FNW(95)) THEN L=9:
   ENDPROC
3140 IF d$=FNX(FNW(96)) OR d$=FNX
   (FNW(97)) THEN L=1Ø:ENDPROC
3150 IF d$=FNX(FNW(98)) THEN L=15:
   ENDPROC
3160 IF d$=FNX(FNW(99))THEN L=21:
   ENDPROC
3170 IF d$=FNX(FNW(1ØØ)) THEN L=11:
   ENDPROC
3180 IF d$=FNX(FNW(1Ø1)) OR d$=FNX
   (FNW(1Ø2)) THENL=2Ø:ENDPROC
3190 IF d$=FNX(FNW(1Ø3)) OR d$=FNX
   (FNW(1Ø4)) THEN L=17:ENDPROC
3200 IF d$=FNX(FNW(1Ø5))OR d$=FNX
   (FNW(1Ø6)) THEN L=3:ENDPROC
3210 IF d$=FNX(FNW(1Ø7)) THEN L=12:
   ENDPROC
3220 IF d$=FNX(FNW(1Ø8)) OR d$=FNX
   (FNW(1Ø9))THEN L=13:ENDPROC
3230 IFd$=FNX(FNW(11Ø)) THEN L=5:
   ENDPROC
3240 PRINT"I don't know where the "d$
   "is.":INPUT"Try again"d$:GOTO 3Ø9Ø
```

```
1780 PRINT@264,"";:WN=65:GOSUB51ØØ
1800 PRINT:WN=33:GOSUB51ØØ
1810 FOR DU=1 TO 1ØØØ:IF INKEY$=""
   THEN 181Ø
1820 GOTO 184Ø
1830 RETURN
1840 REM *** Proc b
1850 CLS
1860 PRINT@266,"YOU'RE DEAD!"
1870 PRINT@298,"=========
   ==="
1880 GOTO 65ØØ
1890 CLS
1900 IF K(3)<>-1 THEN PRINT:WN=66:
   GOSUB51ØØ:PRINT:WN=67: GOSUB
   51ØØ:EXEC41194:RETURN
1910 IF K(2)<>-1 THEN PRINT:WN=68:
   GOSUB51ØØ:PRINT:WN=67: GOSUB
   51ØØ:EXEC41194:L=L-6:RETURN
1920 PRINT@267,"WELL DONE!"
1940 PRINT@299,"YOU'VE WON"
1960 GOTO 65ØØ
1970 REM *** Proc c
1980 Q7=Ø:FOR C7=1 TO 21
1990 IF INSTR(Ø$(C7),T$)>Ø THEN
   Q7=C7
2000 NEXT
2010 IF Q7=Ø THEN PRINT"I DON'T
   UNDERSTAND□";T$:GOTO2Ø7Ø
2020 IF K(Q7)=-1 THEN PRINT"YOU'VE
   ALREADY GOT IT!":GOTO2Ø7Ø
2030 IF K(Q7)<>L THEN PRINT"THE□";
   T$;"□ISN'T HERE!":GOTO2Ø7Ø
2040 IF P7>3 THEN PRINT"YOU CAN'T
   CARRY ANY MORE":GOTO2Ø7Ø
2050 K(Q7)=-1:P7+1:PRINT"OK-
   YOU'VE GOT IT"
2060 WN=166:GOSUB52ØØ:IF T$=Z$
   THEN X7=X7+6ØØ
2070 GOSUB55ØØ
2080 RETURN
2090 REM *** Proc d
2100 WN=153:GOSUB52ØØ:D1$=Z$:
   WN=155:GOSUB52ØØ:IF E$(L)=D1$
   OR E$(L)=Z$ THEN WN=69:GOSUB
   5ØØØ:PRINTZ$,E$(L);"!":WN=34:GOSUB
   51ØØ:GOSUB55ØØ:GOTO65ØØ
2110 IF E$(L)="" THEN WN=35:GOSUB
   51ØØ:GOSUB55ØØ:GOSUB55ØØ:GOTO 244Ø
2120 WN=129:GOSUB52ØØ:IF E$(L)=Z$
   THEN WN=36:GOSUB51ØØ:GOSUB55ØØ:
   GOSUB55ØØ:GOTO244Ø
2130 C$="":H$=""
2140 W7=1:A7=Ø
2150 IF K(2Ø)=-1 THEN W7=W7+2
2160 IF K(9)=-1 THEN W7=W7+3
2170 IF K(15)=-1 THEN A7=A7+.5
2180 IF K(14)=-1 THEN A7=A7+.5
2190 IF K(1Ø)=-1 THEN W7=W7+1
2200 IFA7=1 THEN W7=W7+3
2210 IF W7=1 AND C$="" THEN INPUT
   "FIGHT WITH BARE HANDS (Y/N)";H$
2220 WN=7Ø:GOSUB52ØØ:IF E$(L)=Z$
   AND H$="Y" THEN WN=37:GOSUB
   51ØØ:GOSUB55ØØ:GOTO184Ø
2230 WN=135: GOSUB52ØØ:IF H$="Y"
   AND F(L)>1 AND E$(L)<>Z$ THEN
   PRINT"YOU CAN'T FIGHT THE□"; E$(L):
   PRINT"WITH BARE HANDS!":
   GOSUB55ØØ:GOTO244Ø
2240 IF H$<>"Y" AND W7=1 THEN 244Ø
2250 E7=RND(6):CLS
2260 CLS8
2290 PRINT@268,"fighting";
2300 SCREENØ,1:PRINT@48Ø,"";
2310 GOSUB55ØØ
```

```
2320 IF W7>F(L) AND E7>2 THEN
   V=V-2:GOTO2360
2330 IF W7>F(L) AND E7<=2 THEN
   V=V-1:GOTO2380
2340 IF W7<=F(L) AND E7>=4 THEN
   V=V-3:GOTO2360
2350 IF W7<=F(L) AND E7<4 THEN
   V=V-3:GOTO2380
2360 IF V<1 THEN 1840
2370 PRINT:PRINT"YOU ARE WOUNDED!"
   :PRINT"YOUR VITALITY IS";V:GOTO2410
2380 IF V<1 THEN 1840
2390 PRINT:PRINT"YOU HAVE WON THE
   BATTLE!":PRINT"YOUR VITALITY IS";V
2400 E$(L)="":GOSUB5500:RETURN
2410 L7=RND(21):IF K(L7)=-1 THEN
   PRINT"YOU HAVE DROPPED THE□";
   O$(L7):K(L7)=L:P7=P7-1
2420 INPUT "CONTINUE THE FIGHT (Y/N)";C$
2430 IF C$="Y" THEN 2140
2440 CLS:RETURN
2450 REM *** Proc e
2460 CLS
2470 PRINT"YOU HAVE COLLECTED:-":P7=Ø
2480 FOR C7=1 TO 21
2490 IF K(C7)=-1 THEN PRINT"THE□";
   O$(C7):P7=P7+1
2500 NEXT
2510 IF P7=Ø THEN PRINT"NOTHING"
2520 PRINT"YOUR VITALITY IS";V
2530 GOSUB5500:GOSUB5500
2540 RETURN
2550 REM *** Proc f
2560 CLS
2570 WN=71:GOSUB5100
2580 GOSUB5500
2590 J=RND(6)
2600 IF T7<1 AND I7<1 THEN WN=72:
   GOSUB5100:GOTO2720
2610 IFT7=1ANDI7=1ANDJ>3THEN2680
2620 IFT7<1ANDI7=1THEN2680
2630 WN=38:GOSUB5100
2640 GOSUB5500:GOSUB5500:T7$=""
2650 FOR C7=Ø TO 5:Q7=RND(26)+64:
   T7$=CHR$(Q7)+T7$:NEXT
2660 WN=73:GOSUB5100:PRINTT7$:T7=Ø
2670 GOSUB5500:GOTO2730
2680 WN=39:GOSUB5100
2690 GOSUB5500:GOSUB5500:I7$=""
2700 FOR C7=Ø TO 5:Q7=RND(26)+64:
   I7$=CHR$(Q7)+I7$:NEXT
2710 WN=73:GOSUB5100:PRINTI7$:I7=Ø
2720 GOSUB5500
2730 CLS:RETURN
2740 REM *** Proc g
2760 CLS:WN=74:GOSUB5100
2770 E$(L)=JM$:F(L)=1Ø
2780 N=Ø:S=Ø:E=Ø:W=Ø:U=Ø:D=Ø:
   F=1
2790 RETURN
2800 REM *** Proc h
2810 IF E$(L)="" THEN WN=75:GOSUB
   51ØØ:GOSUB55ØØ:GOTO298Ø
2820 WN=153:GOSUB52ØØ:D1$=Z$:
   WN=155:GOSUB52ØØ:IF E$(L)=D1$
   OR E$(L)=Z$ THEN PRINT"NO DEAL!":
   WN=34:GOSUB51ØØ:GOSUB55ØØ:GOTO
   65ØØ
2830 PRINT@416,"WHAT'S YOUR OFFER";:
   INPUTOF$:VA=Ø
2840 WN=166:GOSUB52ØØ:IF K(21)=-1
   AND OF$=Z$ THEN 289Ø
2850 WN=166:GOSUB52ØØ:IF OF$=Z$
   THEN WN=76:GOSUB51ØØ:GOSUB55ØØ:
   RETURN
2860 WN=126:GOSUB52ØØ:D1$=Z$:
   WN=77:GOSUB52ØØ:D2$=Z$:WN=7Ø:
   GOSUB52ØØ:IF OF$=D1$ OR OF$=
   D2$ AND K(1)=-1 AND E$(L)=Z$
   THEN PRINT"IT'S A DEAL":GOSUB55ØØ:
   E$(L)="":K(1)=Ø:RETURN
2870 IF OF$=D1$ AND E$(L)=Z$ THEN
   WN=78:GOSUB51ØØ:GOSUB55ØØ:GOTO
   53Ø
2880 WN=166:GOSUB52ØØ:IF OF$<>Z$
   THEN WN=79:GOSUB5ØØØ:PRINTZ$;
   OF$;"?":GOSUB55ØØ:GOTO298Ø
2890 PRINT"YOU HAVE";X7;" GOLD":
   WN=8Ø:GOSUB5ØØØ:PRINTZ$;:INPUT OF
2900 IF OF>X7 THENWN=81:GOSUB51ØØ:
   GOTO289Ø
```



```
2910 CLS:WN=82:GOSUB51ØØ:GOSUB55ØØ
2920 PR=RND(12)*5Ø
2930 IF PR>OF THEN CLS:WN=83:GOSUB
   5ØØØ:PRINTZ$:WN=84:GOSUB5ØØØ:
   PRINTZ$;:INPUTIN$
2940 IF OF>=PR THEN299Ø
2950 IF IN$="Y" THEN289Ø
2960 WN=129:IF IN$<>"Y" AND E$(L)
   <>Z$ THEN WN=85:GOSUB51ØØ:
   GOSUB55ØØ:GOSUB2Ø9Ø:RETURN
2970 IF IN$<>"Y" THEN WN=86:GOSUB
   51ØØ:GOSUB55ØØ:PRINT"YOU
   SURRENDER":GOSUB55ØØ:GOTO 65ØØ
2980 CLS:RETURN
2990 PRINT"OK - IT'S A
   DEAL":E$(L)="":X7=X7-OF
3000 IF X7=Ø THENK(21)=21
3010 IFOF<>Ø THEN PRINT "YOU'VE
   LOST";OF,"□GOLD"
3020 GOSUB55ØØ
3030 RETURN
3040 REM***Proc i
3050 CLS
3060 WN=87:GOSUB51ØØ
3070 IFI$=I7$ THEN 325Ø
3080 IF I$=T7$ AND T7=Ø THEN
   PRINT:INPUT "WHERE DO YOU WANT TO
   GO";D7$:T7=-1
3090 WN=88:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=19:RETURN
3100 WN=9Ø:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=1:RETURN
3110 WN=92:GOSUB52ØØ:IF D7$=Z$
   THEN L=8:RETURN
3120 WN=93:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=2:RETURN
3130 WN=95:GOSUB52ØØ:IF D7$=Z$
   THEN L=9:RETURN
3140 WN=96:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=1Ø:RETURN
3150 WN=98:GOSUB52ØØ:IF D7$=Z$
   THEN L=15:RETURN
3160 WN=99:GOSUB52ØØ:IF D7$=Z$
   THEN L=21:RETURN
3170 WN=1ØØ:GOSUB52ØØ:IFD7$=Z$
   THENL=11:RETURN
3180 WN=1Ø1:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=2Ø:RETURN
3190 WN=1Ø3:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=17:RETURN
3200 WN=1Ø7:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=3:RETURN
3210 WN=1Ø7:GOSUB52ØØ:IFD7$=Z$
   THENL=12:RETURN
3220 WN=1Ø8:GOSUB53ØØ:IF D7$=D1$
   OR D7$=Z$ THEN L=13:RETURN
3230 WN=11Ø:GOSUB52ØØ:IF D7$=Z$
   THEN L=5:RETURN
3240 PRINT"I DON'T KNOW WHERE THE
   ";D7$:INPUT "IS, TRY
   AGAIN";D7$:GOTO3Ø9Ø
```

CUMULATIVE INDEX

An interim index will be published each week. There will be a complete index in the last issue of *INPUT*.

COMING IN ISSUE 48...

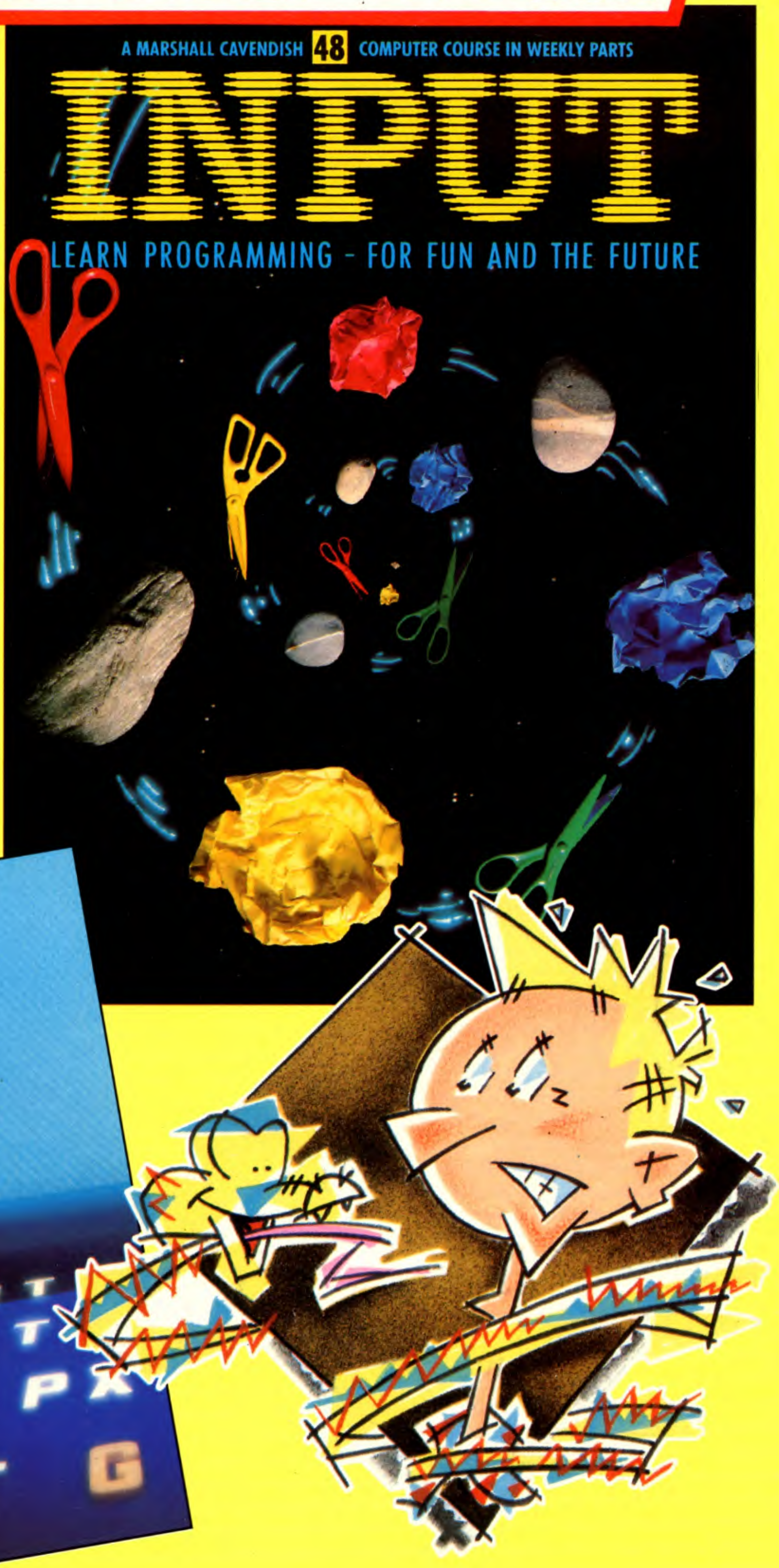
❑ *Try to outbluff your computer. No matter how hard you try, when playing SCISSORS, PAPER, STONE, it'll see right through you by making fast statistical calculations*

❑ *Make life easier for yourself when developing and debugging programs. Our PROGRAM CROSS-REFERENCER is a handy utility which will list selected lines and search for and replace variable names*

❑ *Go FORTH and write stacks of programs after you've found out how to structure this fast and efficient general purpose language*

❑ *Try out some serpent training, or spend a night on the reptiles in Cliffhanger. Add routines to SHAKE THOSE SNAKES and poke their tongues*

❑ *. . . and continue with ESCAPE*



ASK YOUR NEWSAGENT FOR INPUT